Use of Tabu Search in a Solver to Map
Complex Networks onto Emulab Testbeds

THESIS

Jason E. MacDonald, Captain, USAF

AFIT/GCE/ENG/07-07

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCE/ENG/07-07

Use of Tabu Search in a Solver to Map Complex Networks onto Emulab Testbeds

THESIS

Presented to the Faculty
Department of Electrical and Computer Engineering
Graduate School of Engineering and Management
Air Force Institute of Technology
Air University
Air Education and Training Command
In Partial Fulfillment of the Requirements for the
Degree of Master of Science in Computer Engineering

Jason E. MacDonald, B.S.E.E.
Captain, USAF

March 2007

AFIT/GCE/ENG/07-07

Use of Tabu Search in a Solver to Map Complex Networks onto Emulab Testbeds

Jason E. MacDonald, B.S.E.E.
Captain, USAF

Approved:

Dr. Robert F. Mills (Chairman)	13 MAR 07 date
Dr. Rusty O. Baldwin (Member)	13 Mar 07 date
Dr. Barry E. Mullins (Member)	13 Mar 07 date

AFIT/GCE/ENG/07-07

Abstract

The University of Utah's solver for the testbed mapping problem, *assign*, uses a simulated annealing metaheuristic algorithm to map a researcher's experimental network topology onto available testbed resources. This research modifies *assign* to use tabu search to find near-optimal physical topology solutions to user experiments consisting of random and scale-free complex networks. Complex networks often have hundreds or thousands of nodes and are used to describe large complicated systems ranging from genetics to the Internet. While simulated annealing arrives at solutions almost exclusively by chance, tabu search incorporates the use of memory and other techniques to guide the search towards good solutions. Both versions of *assign* are compared to determine whether tabu search can produce equal or higher quality solutions than simulated annealing in a shorter amount of time. It is assumed that all testbed resources remain available, and that hardware faults or another competing mapping process do not remove testbed resources while either version of *assign* is executing. The results show that tabu search is able to produce a higher proportion of valid solutions for 34 out of the 38 test networks than simulated annealing. For cases where a valid solution was found, tabu search executes more quickly than simulated annealing for scale-free networks and networks with less than 100 nodes. Simulated annealing is able to produce equal or higher quality solutions for all test networks when a valid solution was found.

Acknowledgements

I am ever thankful to Dr. Bob Mills for his many roles of thesis advisor, life counselor, and research motivator. He played them all well, and often simultaneously. A few minutes of discussion with him would bring into focus hours or days of confusion. I have never met anyone so versed in such a wide array of subjects, and I hope to get the chance to explore some of the many topics we've spoken about. I am grateful for his trust in my abilities, which was often greater than my own self-confidence.

I would like to thank my committee members Dr. Rusty Baldwin and Dr. Barry Mullins for providing many insights and a deep appreciation for the inner workings of computer networks. I would also like to thank Mr. Robert Ricci at the University of Utah for taking the time to review and comment on my ideas for modifying *assign*. I always felt more certain that what I was attempting was possible after his reassurances.

I am deeply indebted to David and Myrna Montminy for helping me make it through AFIT. Being out of school for seven years and changing degrees certainly did not make navigating an already challenging curriculum any easier. David, thank you for helping me through our classes and putting up with my dumb questions. Myrna, thank you for being the best friend anyone could ever ask for. Our PF Chang dinners and Starbuck coffee breaks were pivotal in reconstituting me to take on another day at AFIT. I don't know how I would've done without you guys. I wish the two of you the best of luck in New Mexico.

Lastly, I would like to thank my mother for being my greatest inspiration. I believe it was her strong work ethic and other traits instilled in me over the years that enabled me to achieve as much as I did during graduate school.

Jason E. MacDonald

Table of Contents

Page

Abstract iv

Acknowledgements v

List of Figures viii

List of Tables x

List of Abbreviations xii

I. Introduction 1
- 1.1 Purpose and Goals 4
- 1.2 Assumptions and Scope 6
- 1.3 Organization 7

II. Background 8
- 2.1 Complex Networks 8
  - 2.1.1 Random Networks 9
  - 2.1.2 Small-World Networks 11
  - 2.1.3 Scale-Free Networks 12
- 2.2 Emulation 15
  - 2.2.1 Growth of Network Simulation as a Research Tool 15
  - 2.2.2 Overview of Network Emulation 18
  - 2.2.3 University of Utah's Emulab 19
- 2.3 Metaheuristic Algorithms 23
  - 2.3.1 Simulated Annealing 25
  - 2.3.2 Assign - Emulab's Solver to the Testbed Mapping Problem 25
  - 2.3.3 Improvements to Assign 28
  - 2.3.4 Tabu Search 31
- 2.4 Summary 34

III. Research Methodology 35
- 3.1 Problem Definition 35
  - 3.1.1 Goals and Hypothesis 36
  - 3.1.2 Approach 37
- 3.2 System Boundaries 38
- 3.3 System Services 41

Page

3.4 Workload 42
3.5 Performance Metrics 44
3.6 Parameters 46
3.6.1 System 46
3.6.2 Workload 48
3.7 Factors 48
3.8 Evaluation Technique 49
3.9 Experimental Design 50
3.10 Implementation 51
3.10.1 Original Anneal.cc using SA Search Algorithm . 51
3.10.2 Modified Anneal.cc using TS Search Algorithm 53
3.11 Summary 55

IV. Data Analysis 56
4.1 Validation 56
4.2 Vlink Multiplexing Issues 56
4.3 Analysis of Valid Solutions 60
4.4 Analysis of Execution Time and Solution Quality 65
4.4.1 Analysis of Execution Time 69
4.4.2 Analysis of Objective Function Score 71
4.5 Summary 74

V. Conclusions 75
5.1 Research Summary and Conclusions 75
5.2 Research Significance and Contributions 76
5.3 Future Work 76

Appendix A. Makefile Used to Compile Assign Source Code 77

Appendix B. Virtual Topology and Testbed Resource Input Files . . . 80
B.1 Set of Available Testbed Resources (.ptop file) 80
B.2 SF Virtual Topologies (.top files) 82

Appendix C. Data 89

Bibliography 128

List of Figures

Figure		Page
2.1	Complex Networks Found in Society and Nature	10
2.2	Differences Between a Regular Lattice and a Random Graph	11
2.3	Examples of a Random Graph.	11
2.4	Transition from Regular to Random Graph Using WS Model	12
2.5	Differences Between Random and Scale-free Networks.	13
2.6	Exponential and Scale-free Network Attack Tolerance.	16
2.7	University of Utah Emulab Classic Testbed	19
2.8	Example Vnodes and Pnodes Descriptions.	22
2.9	Landscape of an Objective Function	25
2.10	Migrating Across Non-solution Space to a Lower Minima.	28
3.1	Emulab System Architecture.	39
3.2	System Boundaries of *Assign*.	40
3.3	Diagram of the Synthetic Testbed.	43
3.4	Visualization of the 500-node SF Virtual Topology.	45
4.1	Diagram of the 10-node SF Physical Topology After Mapping.	57
4.2	Three Examples of Vlink Multiplexing.	58
4.3	Valid Solutions for Networks With Less Than 100 Nodes.	63
4.4	Valid Solutions for Networks With 100 Nodes and Greater.	64
4.5	Measured Execution Time by Search Algorithm and Virtual Topology.	66
4.6	Measured Objective Function Score by Search Algorithm and Virtual Topology.	66
4.7	Residual Plots for Simulated Annealing Execution Time.	67
4.8	Residual Plots for Simulated Annealing Objective Function Score.	67
4.9	Residual Plots for Tabu Search Execution Time.	68
4.10	Residual plots for Tabu Search Objective Function Score.	68

Figure Page

4.11 Confidence Interval Plot of Execution Time for All Virtual Topologies. 69

4.12 Confidence Interval Plots of Execution Time by Virtual Topology Type. 70

4.13 Confidence Interval Plot of Execution Time for Virtual Topologies With Less Than 100 Nodes. 70

4.14 Confidence Interval Plot of Execution Time for Virtual Topologies With 100 Nodes and Greater. 71

4.15 Confidence Interval Plot of Objective Function Score for All Virtual Topologies. 72

4.16 Confidence Interval Plots of Objective Function Score by Virtual Topology Type. 72

4.17 Confidence Interval Plot of Objective Function Score for Virtual Topologies With Less Than 100 Nodes. 73

4.18 Confidence Interval Plot of Objective Function Score for Virtual Topologies With 100 Nodes and Greater. 73

List of Tables

Table		Page
2.1	Physical Resources Available in Emulab.	27
3.1	The Number of Gigabit Plinks Per PC Node.	44
3.2	*Assign* Factors and Levels.	48
3.3	*Assign's* Hardware and Software Testing Environment.	49
3.4	*Assign* Runtime Options.	50
4.1	Nine SF Virtual Topologies Modified to Pass *Assign* Prechecks.	59
4.2	Valid Solutions Created for Random Virtual Topologies.	61
4.3	Valid Solutions Created for SF Virtual Topologies.	62
B.1	BRITE Parameters Used to Create the Virtual Topologies.	84
B.2	Modifications Made to BRITE SF Virtual Topologies.	86
B.3	Modifications Made to BRITE Random Virtual Topologies.	87
C.1	Data for 10-node Random Virtual Topology.	90
C.2	Data for 20-node Random Virtual Topology.	91
C.3	Data for 30-node Random Virtual Topology.	92
C.4	Data for 40-node Random Virtual Topology.	93
C.5	Data for 50-node Random Virtual Topology.	94
C.6	Data for 60-node Random Virtual Topology.	95
C.7	Data for 70-node Random Virtual Topology.	96
C.8	Data for 80-node Random Virtual Topology.	97
C.9	Data for 90-node Random Virtual Topology.	98
C.10	Data for 100-node Random Virtual Topology.	99
C.11	Data for 200-node Random Virtual Topology.	100
C.12	Data for 300-node Random Virtual Topology.	101
C.13	Data for 400-node Random Virtual Topology.	102
C.14	Data for 500-node Random Virtual Topology.	103

Table		Page

Table		Page
C.15	Data for 600-node Random Virtual Topology.	104
C.16	Data for 700-node Random Virtual Topology.	105
C.17	Data for 800-node Random Virtual Topology.	106
C.18	Data for 900-node Random Virtual Topology.	107
C.19	Data for 1000-node Random Virtual Topology.	108
C.20	Data for 10-node Scale-free Virtual Topology.	109
C.21	Data for 20-node Scale-free Virtual Topology.	110
C.22	Data for 30-node Scale-free Virtual Topology.	111
C.23	Data for 40-node Scale-free Virtual Topology.	112
C.24	Data for 50-node Scale-free Virtual Topology.	113
C.25	Data for 60-node Scale-free Virtual Topology.	114
C.26	Data for 70-node Scale-free Virtual Topology.	115
C.27	Data for 80-node Scale-free Virtual Topology.	116
C.28	Data for 90-node Scale-free Virtual Topology.	117
C.29	Data for 100-node Scale-free Virtual Topology.	118
C.30	Data for 200-node Scale-free Virtual Topology.	119
C.31	Data for 300-node Scale-free Virtual Topology.	120
C.32	Data for 400-node Scale-free Virtual Topology.	121
C.33	Data for 500-node Scale-free Virtual Topology.	122
C.34	Data for 600-node Scale-free Virtual Topology.	123
C.35	Data for 700-node Scale-free Virtual Topology.	124
C.36	Data for 800-node Scale-free Virtual Topology.	125
C.37	Data for 900-node Scale-free Virtual Topology.	126
C.38	Data for 1000-node Scale-free Virtual Topology.	127

List of Abbreviations

Abbreviation		Page
Emulab	Emulation Laboratory	2
AFIT	Air Force Institute of Technology	2
PCs	Personal Computers	2
GUI	Graphical User Interface	3
NS2	Network Simulator 2	3
WWW	World Wide Web	9
SF	Scale-free	9
ER	Erdos and Renyi	9
WS	Watts and Strogatz	12
U.S.	United States	12
BA	Barabasi and Albert	13
CAT5	Category 5	18
CORE	CyberOpeRations Emulator	20
vclass	virtual equivalence class	22
RAM	Random Access Memory	23
OS	Operating System	23
SA	Simulated Annealing	25
pclass	physical equivalence class	29
LAN	Local Area Network	30
TS	Tabu Search	31
SUT	System Under Test	39
CUT	Component Under Test	40
BRITE	Boston Representative Internet Topology gEnerator	43
ANOVA	Analysis of Variance	65

Use of Tabu Search in a Solver to Map Complex Networks onto Emulab Testbeds

I. Introduction

Complex networks are presently a popular topic of study for researchers across a wide range of seemingly unrelated disciplines. Behaviorial science, molecular biology, medicine and computer science are just a few of the fields in which these networks are under investigation. Networks are a pervasive phenomenon throughout our world, and can be found in human societies, our natural surroundings and even in technology. Whether it is a collection of biological cells connected by axons, molecules connected by biochemical reactions, Hollywood actors associated by the movies they have starred in, or people linked by social relationships, these are all examples of complex networks that share a common architecture and self-organization characteristics. Discovery of the properties of a complex network in one field can lead to breakthroughs and applications in many other branches of science. For instance, a key development in genetics can have impacts on Internet security and terrorist social networks [3, 4, 32].

Nodes in complex networks are made of many non-identical, diverse elements. Links account for many different types of interactions between nodes [3]. Non-random distribution of connectivity among nodes, associativity among nodes, node clustering and evidence of a hierarchical structure are key trademarks of complex networks. The number of nodes in a complex network is usually large, sometimes totaling hundreds, thousands or even millions. The topology of complex networks can change over time, as nodes are added and removed [28, 34].

Mathematical modeling, simulation and direct measurement are the three most common methods for analyzing performance, ascertaining characteristics and predicting responses of networks. Direct measurement is the most straightforward of the

three, and involves taking measurements of the network in question. When the network under test cannot be accessed or does not exist, analytic performance evaluation and simulation make use of models. A drawback of models is they must be, or have been, validated to ensure the model accurately represents the system's behavior [2].

Emulation is a lesser known, but just as important member of the suite of performance evaluation techniques. Emulation combines the use of real elements of the network under test along with simulated or abstracted elements in the same experiment. As a result, emulation can achieve a higher degree of realism than simulation alone. A consequence of increased realism is that the experiment executes in real time to allow simulated and real elements to communicate with each other. Another disadvantage is the inability to completely repeat the series of events that occurred in a given experiment due to the inclusion of real components in the experimental network. Simulation is preferred if the experiment duration is limited to a fraction of the necessary completion time or absolute repeatability is required. The number of real and abstracted components selected for the experiment is dependent on the researcher's demands and available testbed resources [11, 12].

In the academic and research community, University of Utah's Emulation Laboratory (Emulab) testbed is in the forefront among network emulation testbeds. Emulab has been in production use since April 2000, and as of January 2007 the Air Force Institute of Technology (AFIT) and sixteen other universities have constructed testbeds that use Emulab's software environment. The University of Utah's Emulab testbed is composed of three smaller testbeds: a mobile wireless laboratory, a fixed 802.11 wireless testbed and the Emulab Classic testbed. Emulab Classic is "a time- and space-shared 'cluster testbed' whose main goals are to provide artifact-free network emulation for arbitrary experiments, while making them as easy and quick as simulation" [24]. Emulab Classic comprises a host of personal computers (PCs) that communicate with each other by hardwired Fast Ethernet interfaces and network switches. The testbed is space-shared because it can execute multiple experiments simultaneously, whether it be multiple researchers each submitting a single experi-

ment or a single researcher running multiple instances of the same experiment. The testbed is also time-shared because submitted experiments are “swapped out” after a given amount of idle time, allowing other experiments to make use of the resources originally assigned to the first experiment [24, 29].

In the Emulab environment, the term “virtual” refers to components in the researcher’s experiment. Virtual can also refer to simulated resources in the testbed environment. This thesis uses the first definition of virtual unless otherwise noted. The researcher’s submitted topology is known as the *virtual topology*. Nodes and links in the virtual topology are known as a *vnodes* and *vlinks*, respectively. The testbed representation of the researcher’s experiment is known as the *physical topology*. Nodes and links in the physical topology are known as a *pnodes* and *plinks*, respectively.

A researcher builds an experiment script using a Java™ graphical user interface (GUI) accessible on Emulab’s homepage or using the Network Simulator 2 (NS2) program [13, 29]. NS2 is a popular open-source discrete event simulator widely used for networking research. Once the experiment is submitted, Emulab’s software environment “maps” available testbed resources to the researcher’s virtual topology, using a solver known as *assign*. *Assign* has five goals when it maps testbed resources to a virtual topology [14, 24]:

1. Correctly assign vnodes and vlinks to available pnodes and plinks by ensuring specified hardware, software and protocol configurations are met and no artifacts are introduced into the physical topology.

2. Map vlinks to plinks in such a way that inter-switch bandwidth in the physical topology is minimized.

3. Complete the mapping in such a way to maximize the number of experiments that can be run simultaneously on the testbed.

4. Facilitate experiment scaling by minimizing the number of pnodes required for each experiment. This is done by assigning multiple similarly-configured vnodes to a single pnode.

5. Complete the assignment process in a minimal amount of time, much lower than topology creation time to expedite experiment turnaround time.

1.1 Purpose and Goals

Communications infrastructure, networks in military combat performance scenarios [7], command and control electronic mail systems [16] and networks described in network-centric warfare doctrine [6] are just a few examples of defense complex networks. Changing or modifying these networks once deployed can be a laborious operation, as user downtime can be difficult to obtain and migration of existing users to a new infrastructure or process can be tedious and cumbersome. In addition, critical communication networks must be returned to operational status within a moment's notice in case of emergency. This causes project delays since downtime must be renegotiated between operational and support communities. Emulation is one way of identifying weaknesses and vulnerabilities in defense networks prior to their exploitation and without disruption of real-world operations. In this way, the ability to protect critical communication channels in times of devastation and war is enhanced. During normal operations, studying the performance of these networks can also lead to increased reliability, availability and user confidence.

Mr. Mike Hibler, Mr. Robert Ricci and other key architects of Emulab state in [14] that a primary challenge of emulation environments is scale. As studies into complex networks increase, emulation testbeds will need to be able to support networks with larger numbers of nodes. The initial version of the Emulab environment conservatively mapped virtual components one-to-one onto their physical counterparts to meet the first goal of mapping the virtual topology without experimental artifacts. It is no longer sufficient to map vnodes and vlinks one-to-one onto testbed pnodes and plinks. To prevent a large experiment from monopolizing an entire testbed and to facilitate study of networks with thousands and tens of thousands of nodes, multiplexing more than one vnode or vlink onto a single pnode or plink must be supported [14,24].

Complete accuracy is not always required in many cases of academic research and performance analysis. High fidelity evaluations are often inefficient and slow down development time, and often the only the systems characteristics that need to be modeled are the ones related to the research [2]. To support a greater degree of multiplexing, later Emulab environments have relaxed the first goal of the testbed mapping algorithm. A higher degree of multiplexing improves the efficiency of mapped components. For example, vlinks rarely make use of their maximum allocated bandwidth so grouping multiple vlinks wastes less bandwidth in the underlying plink [14].

Mapping a thousand or ten thousand node virtual topology represents a challenge in Emulab. The fifth goal of the testbed mapping algorithm is to complete the assignment process in a minimal amount of time. Due to the dynamic nature of the Emulab testbed environment, multiple researchers may be attempting to submit different experiments simultaneously. If the mapping process on a given virtual topology takes too much time to complete, some of the testbed resources that it chose may no longer be available, forcing the algorithm to restart [14, 24]. As stated in [24], "Locking experiment creation for hours while large experiments map is not a reasonable solution to this problem." *This research is thus concerned with the problem of creating high quality, feasible solutions for complex networks with thousands of nodes in a minimum amount of time.*

Solution quality is a measurement of how well the resultant physical topology represents the intended test network and the amount of testbed resources used by the physical topology (e.g., how well the experiment was "packed" onto the testbed). *Assign's* first goal of correctly mapping vnodes and vlinks onto testbed resources is of primary importance. However, minimizing experiment instantiation time is more important than saving a few testbed resources, especially if doing so causes mapping time to extend from seconds to minutes or even hours. Observations in [14] note that researchers spend many hours debugging virtual topologies submitted to Emulab.

Therefore, the time it takes to map, or remap, a virtual topology is an important consideration.

Assign uses a simulated annealing metaheuristic algorithm to match testbed resources to a user's virtual topology. One of the ways to reduce the amount of time it takes to instantiate a user experiment on an Emulab testbed is to select another metaheuristic that is able to produce a physical topology specification in less time than simulated annealing. Tabu Search is a metaheuristic that mimics the concept of memory, as opposed to annealing, to solve difficult optimization problems. Memory guides tabu search towards good solutions based on information collected during the search [8, 10]. *The goal of this research is to determine whether a tabu search implementation of* assign *is superior to Emulab's existing simulated annealing implementation in terms of execution time and solution quality.*

1.2 Assumptions and Scope

The process of mapping a researcher's virtual topology to a physical testbed topology has many different phases. The scope of this research is limited strictly to the phase which incorporates the testbed mapping algorithm. The mapping algorithm takes as its input two text files with the extensions .top and .ptop. The .top file is a text file created after the virtual topology produced in the Java™ GUI or NS2 script is parsed into an intermediate format by the Emulab software environment. The .ptop is a text file with available testbed resources for the experiment being submitted. The assumptions for this research are:

- The virtual topology submitted through the Java™ GUI or NS2 script has no syntax errors and is successfully parsed into an intermediate .top file.
- The virtual topology submitted to the testbed mapping algorithm has not been preprocessed in any way to reduce its size and complexity (e.g., graph coarsening).

- A solution in the form of a physical topology can be produced for the submitted virtual topology from the set of available resources specified in the .ptop file, including all special hardware and software requirements.
- The set of available testbed resources specified in the .ptop file does not change while the testbed mapping algorithm is running due to another mapping process allocating resources before the *assign* has completed. Only one instance of *assign* is executing on the set of available of available testbed resources at any given time.
- All testbed resources are operational, working correctly and will not succumb to hardware, software or other faults that may cause the resource to become unavailable.
- Only "cluster" resources from a testbed such as Emulab Classic or the AFIT CORE make up the set of available testbed resources.

1.3 Organization

The remainder of this thesis is divided into four chapters. The next chapter reviews theories surrounding complex networks, introduces the performance evaluation technique of network emulation, and describes metaheuristic algorithms, specifically the simulated annealing and tabu search algorithms. Chapter 3 outlines the research methodology to include the boundaries of the problem, factors selected, performance metrics and experimental design. Chapter 4 analyzes and interprets the data collected. The last chapter concludes with a summary of the research conducted, discusses research significance and contributions, and suggests areas for future research.

II. Background

This chapter is divided into three sections, each covering a main topic of interest. The first portion presents key background information on past and present theories surrounding complex networks. The second introduces the performance evaluation technique known as emulation, and gives an overview of the Emulab testbed environment developed by the University of Utah. The final section contrasts classic iterative searches with metaheuristic search algorithms and outlines the salient features of the simulated annealing and tabu search algorithms. *Assign*, Emulab's solver to the testbed mapping problem, is introduced and improvements to *assign's* performance are discussed.

2.1 Complex Networks

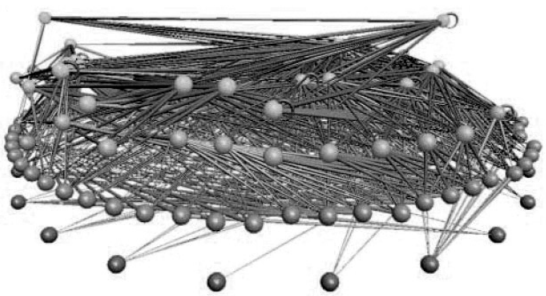
Complex networks are defined as networks with "a non-trivial topological structure" [34] that display certain unique characteristics. The key characteristics of complex networks are [3, 28, 34]:

- Nodes in complex networks are made of many diverse elements. A complex social network can be made up of many different types of people differing in gender, race, and religion. Nodes in a complex biological network can be made up of a wide array of diverse organisms.
- Links can account for many different types of interactions between nodes. In a complex genetic network, links represent a large number of chemical interactions between genes and proteins.
- Complex networks often display a non-uniform distribution of connectivity among nodes. This distribution is due to associativity between nodes or the preference of a large set of nodes to establish links with another, smaller set of nodes.
- Nodes in complex networks often group together to form clusters. These clusters can also group together to form larger clusters, creating hierarchical structures.

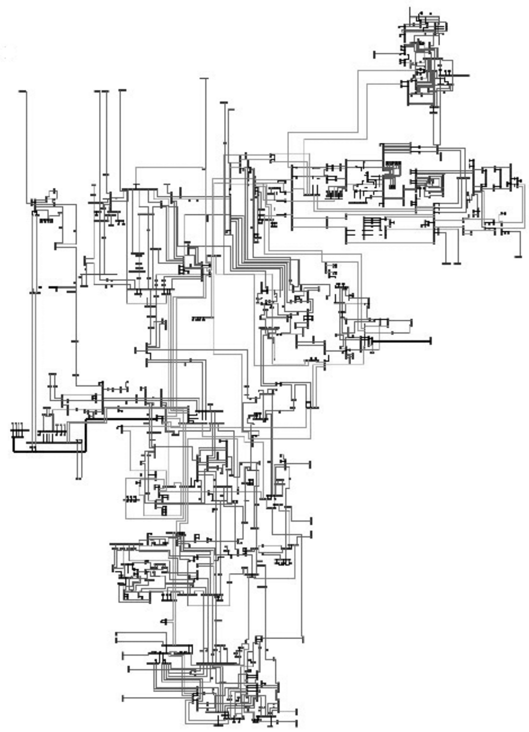
- Complex networks usually have a large number of nodes. The number of nodes can sometimes total in the hundreds, thousands, or even millions.
- The topology of complex networks changes over time. This concept of growth is usually the result of nodes being added or removed from the network.

Complex networks are used to describe numerous systems found throughout nature and society. The routing structure of the Internet is an example of a system that can be represented as a complex network. The nodes of this network are the Internet's tier-1, tier-2 and tier-3 routers and the links are transmission lines (e.g., T-carrier 1 and Optical Carrier 3) that connect them. From a different perspective, the Internet can be described as a complex network by treating webpages that make up the World-Wide Web (WWW) as nodes, and edges as hyperlinks interconnecting them. Complex networks are not confined solely to the domain of computer science. Personnel in a social organization and the relationships that connect them can be expressed as a complex network. In the medical field, the collection of neurons in the brain is a complex network [4, 28, 32]. Figure 2.1 highlights two additional examples. The ecological web of Little Rock Lake is a hierarchical complex network of predator-prey association. The New York State power grid is also a complex network of generators, substations, power lines and transformers. The three leading classifications of complex networks are random, small-world and scale-free (SF) [28, 32, 34].

2.1.1 Random Networks. Prior to the late 1950's, prevailing network theory used geometrically regular graphs (e.g., lattices, chains, or grids) to model processes, relationships, and physical phenomena. This application of simple network theory focused on the nodes or individual network elements, thus investigation of the dynamics of the network as a whole went mostly ignored [28, 32]. In 1959, Paul Erdos and Alfred Renyi introduced the idea of random graphs to represent complex networks. Figure 2.2 shows the differences between a regular lattice graph and a random graph presented by Erdos and Renyi (ER). In the ER graph model, nodes are not



(a)



(b)

Figure 2.1: (a) Food web of Little Rock Lake, Wisconsin. Nodes are functionally distinct "trophic species" and the links show "who eats whom" in the lake.
(b) New York State electric power grid. Nodes are generators and substations represented by small bars. Links are transmission lines and transformers shown by lines interconnecting the generators and substations. Line thickness indicates various voltage levels [28].

fully connected, rather links between neighboring nodes are established with a specified global probability [9].

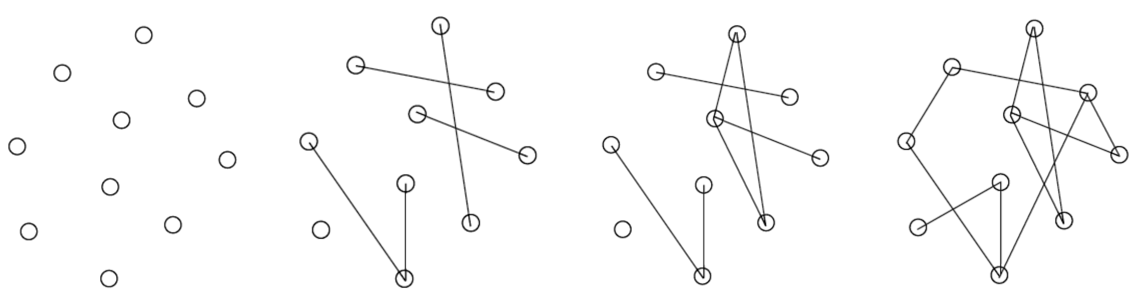
Figure 2.3 shows how the graph connectivity varies with different probability values. In an ER graph, the majority of nodes have the same number of links. It is rare to find nodes that have significantly more or less links than the average, as the number of links per node follows a Poisson or normal (bell-shaped) distribution, as shown in Figure 2.5. The ER network model is very democratic, as each node has approximately the same amount of impact to the overall topology of the network [4]. For the next 40 years, the ER random graph model remained the prominent theory to describe complex networks and their topologies. This was because there were no competing

(a) (b)

Figure 2.2: (a) Graph based on a two-dimensional lattice network in which each node is connected to its nearest neighbor.
(b) Graph based on the ER random model. Each pair of nodes are are connected based on a global probability.

complex network theories that displayed the rigor of the ER model and no capability to map and observe the topology of real-life large complex networks [3, 28, 32]. This changed at the turn of century with the advent of super-computers.

2.1.2 Small-World Networks. In 1998, Duncan Watts and Steven Strogatz found a relationship between geometrically regular graphs and graphs based on the ER model. Their research yielded a model that transitions from a regular to random graph by randomly reassigning links to different nodes with a given probability P. The minimum value of $P = 0$ results in the original geometrically regular graph. The maximum value of $P = 1$ gives a completely random topology, as described



(a) (b) (c) (d)

Figure 2.3: Four examples of a ten-node random graph. The probability that each pair of nodes is connected for each of the cases is (a) 0 (b) 0.1 (c) 0.15 and (d) 0.25 [32].

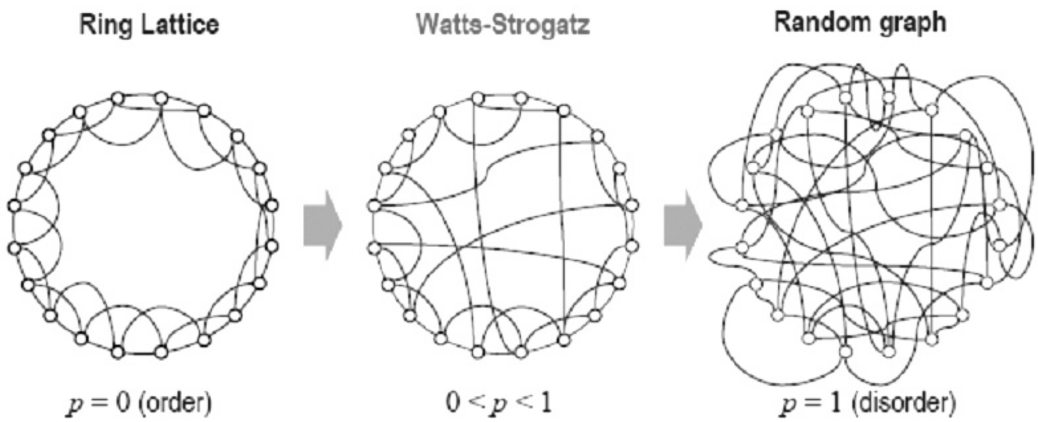
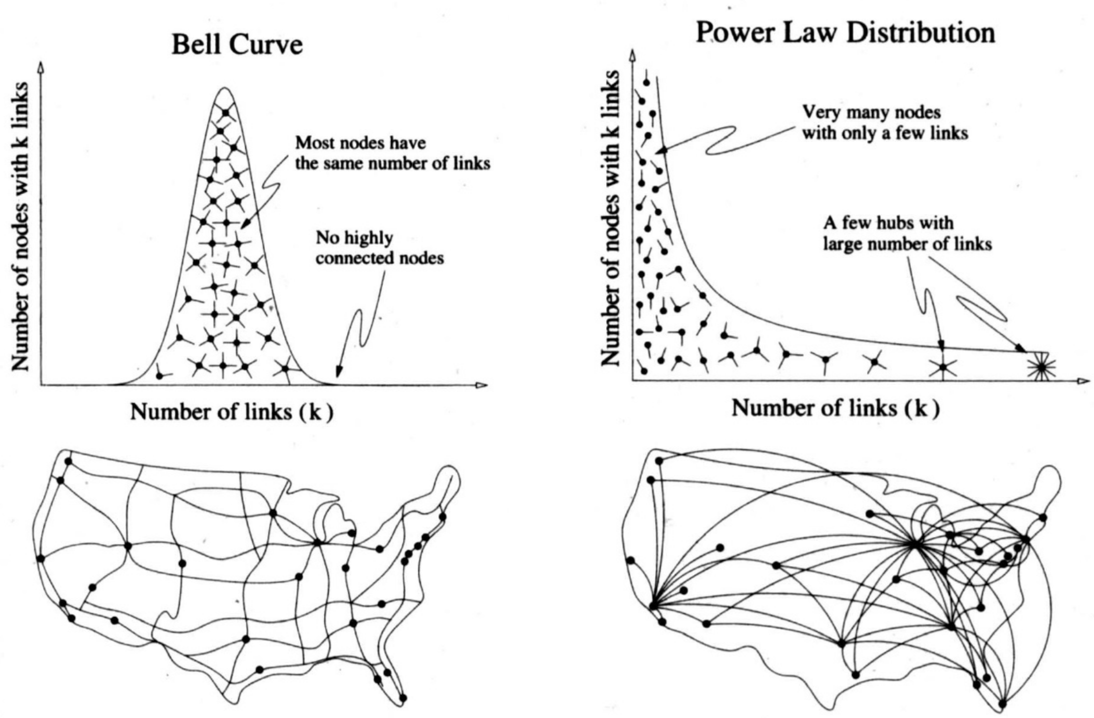


Figure 2.4: The WS model describes how a regular lattice graph transitions to a random graph by varying the probability P from 0 to 1 [32].

by the ER model. As the value of P shifts through the bounds of $0 < P < 1$, a "small-world" graph is created. This small-world characteristic describes how in large complex networks, a random pair of nodes are connected to each other through a relatively short path. In social networks, this is sometimes referred to as "six degrees of separation" [28, 32]. Figure 2.4 shows how the transition from a regular lattice to a random graphs takes place using the Watts and Strogatz (WS) small-world graph model.

2.1.3 Scale-Free Networks. Also in 1998, Albert-Laszlo Barabasi, Eric Bonabeau, Hawoong Jeong and Reka Albert conducted an experiment to map a portion of the WWW. They found that the number of hyperlinks pointing to webpages was not evenly-distributed and did not follow the ER model as hypothesized, rather there were a very few number of webpages that had far more hyperlinks pointing to them than the average. The results of their experiment showed that 80 percent of webpages had fewer than four hyperlinks, while less than 0.01 percent (designated as "hubs") had more than 1,000 [3, 4, 32, 36]. Figure 2.5 shows a case similar to the WWW in which the United States (U.S.) airline system has a few major airports that have a large number of flights to other airports compared to the average, contrasted



(a) (b)

Figure 2.5: (a) The U.S. highway system resembles a random network in which city nodes are randomly connected to each other and all nodes have approximately the same number of links. (b) In contrast, the U.S. airline system is a SF network. Major airport nodes have many links while the majority of airport nodes only have a few connections [4].

against the U.S. highway system which has a more uniformly distributed number of interstate highways between major metropolitan cities.

The existence of nodes with connectivity magnitudes greater than average led Barabasi to coin the phrase "scale-free," due to the number of links per node following a power law distribution (implying lack of scale) and self-similar fractal characteristics. In SF networks, these hubs dramatically influence the way the overall network behaves and operates [28, 34]. The probability $P(k)$ that a given node in a SF network is connected to k other nodes in the Barabasi and Albert (BA) SF graph model is

$$P(k) \sim k^{-\gamma}. \tag{2.1}$$

BA SF networks display many different power-law coefficient γ values, however most fall within the bounds of $2 < \gamma \leq 3$ [3, 4, 36].

The BA SF model incorporates the ideas of growth and preferential attachment to further distinguish SF graphs from random graphs. In contrast to the ER model which assumes the number of nodes in a complex topology is fixed, the BA model incorporates the addition of nodes over time to better represent how a real network expands with age. For example, webpages are constantly being added to the WWW and new routers continually come online to connect to the physical topology of the Internet. Preferential attachment captures the fact that the connectivity of nodes in real networks is not uniform. There are many reasons why certain nodes are preferred over others, including age, reliability, resource availability and location. A newly created webpage will often link with a more well-known and established web portal in order to gain visibility. A recently published paper or article will often prefer to cite an older, reputable publication that has gained acceptance in the community, rather than a new article that has just been printed. The probability that a node will be chosen for attachment increases with its popularity, leading to the "rich get richer" phenomenon. It is these two features of growth and preference that explain why power-law degree distribution is evident in SF networks [3, 4].

Complex networks are present in many military and defense sectors. Many of these networks are thought to be SF, including communication networks, networks in military combat performance scenarios [7], command and control electronic mail systems [16] and networks described in network-centric warfare doctrine [6]. The reliability of SF networks becomes a major concern when operating in such critical functions. SF networks are very robust in situations where nodes are removed randomly from the topology, such as accidental equipment failures in a large communication network or inadvertent cell mutations in the human body as a result of misfolded proteins because random removal of nodes will eliminate mainly nodes that are not hubs due to the "inhomogeneous connectivity distribution" [1] of SF networks. Targeted attacks on hub nodes, however, can have a crippling effect. It has been sug-

gested that eliminating as few as 5 to 15 percent of the hubs in a SF network can cause significant disruptions [4].

The diameter of a network is defined as "the average length of the shortest paths between any two nodes" [1]. The network diameter characterizes network performance by measuring the length of the shortest path used by two nodes to communicate. A smaller diameter indicates a shorter expected path length between any two nodes in the network, resulting in better performance due to reduced latency. Figure 2.6 shows how the performance of a SF network remains relatively unchanged after a random attack, but a targeted attack that removes hub nodes quickly diminishes performance. Exponential networks, conversely, degrade at the same rate after either type of attack.

Whether or not the physical topology of the Internet is a SF network is a subject of much debate. Some claim that the physical layer of the Internet is indeed a SF network and that such a topology represents an "Achilles Heel" [1,4]. In this scenario, a coordinated attack can cause global Internet outages by disabling a small number of key routers. Others argue that the amount of different properties of SF networks is growing due to popularity in recent literature, but none of this literature provides rigorous validation resulting in many contradictions and sensational claims regarding SF networks [18]. No matter which SF network definition is applied to the physical topology of the Internet, reports of the fragile nature of the Internet in the face of a targeted attack are false [18]. Since factors such as design, evolution, functionality and constraints which are ignored in the SF definition. At levels of network abstraction higher than the physical layer of the Internet (i.e., the WWW or electronic mail), SF models may be more appropriate [18].

2.2 Emulation

2.2.1 Growth of Network Simulation as a Research Tool. The use of computer simulation to analyze and predict network performance is quickly becoming widespread in commercial and academic communities. The easy availability of personal computers as research tools have made computer simulation the most common

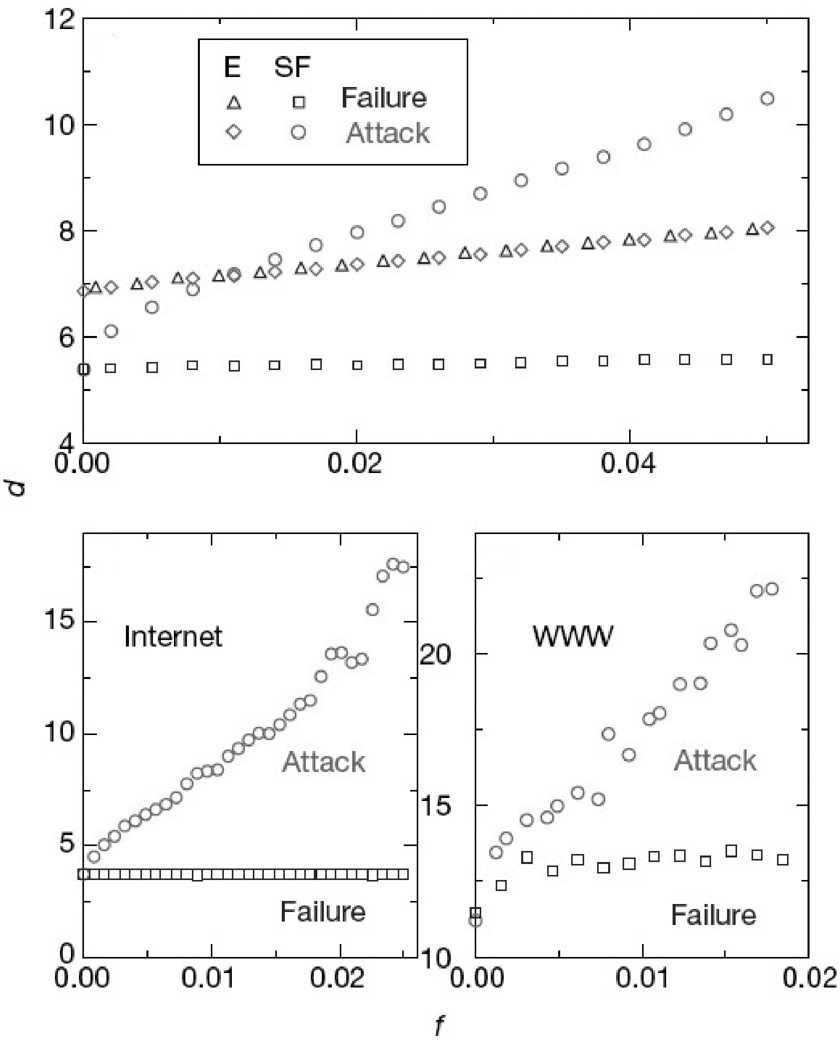


Figure 2.6: The upper pane shows changes in average shortest path length due to accidental and targeted node removal in SF and exponential networks. Both networks contain 10,000 nodes and 20,000 links. Triangles correspond to the diameter of the exponential network as nodes are randomly removed. Squares show the diameter of the SF network when nodes are removed randomly. Diamonds show the response of the exponential network and circles show the response of the SF network to intentional attacks, when the highest connected nodes are removed. The lower left and lower right panes show projected changes as a result of targeted attacks to the physical Internet and WWW topologies, respectively. The responses of these networks correspond to that of a SF topology. [1].

method of scientific investigation [22]. Simulation is prevalent in the telecommunication industry as a foundation to plan for network deployment and other decision support systems [20]. There are 27 different simulation tools in widespread use to-

day [23]. Thus, there is not only a pervasive use of computer simulation tools to study network behavior, but also a wide variety as well.

Of the many network simulation tools available, many are specialized, and all display various strengths and weaknesses. Some of the more common simulation software package weaknesses are:

1. The ability to only model specific classes of network configurations (e.g., wide-area or local-area networks).
2. Lack of user-friendly interfaces.
3. Lack of ability to build custom user models.
4. Overly simple or conversely, overly complex simulation engines.
5. Invalid network models.
6. High cost.

All but a few of the available simulation packages are proprietary and meant to operate in a standalone environment. The few that are interoperable are so with only one other simulation tool and confined to only certain network configurations and models [23]. Surveyed simulation tools cannot generally export user data in a way that is compatible with other standalone simulation software tools, and therefore cannot be used together seamlessly [21, 23]. Instead, a user must work to become an expert in multiple simulation environments to validate results gained from one simulation tool by using another. This is often the only alternative if an analytic model does not exist for the network design being simulated, or if performance is being considered for an analytic model outside the range of system characteristics that make it tractable [25]. The consequence is the tendency of commercial engineers and academic researchers to only use a single simulation software package, and thus make decisions based on results from models that may be overly simplistic, excessively complicated or otherwise biased or corrupt.

2.2.2 Overview of Network Emulation. Emulation is an alterative to the well-known performance evaluation techniques of analytic modeling, simulation and direct measurement of a real system [2]. Emulation gets the experimenter closer than simulation to the responses of an experimental system without actually implementing it. Where simulation makes use of software or other tools to mimic the responses of an experimental system, emulation uses real system components as elements of the experimental system to observe responses the implemented system would have to real-life stimuli. Thus, emulation incorporates more realism into a model than simulation and can be used as an additional method to validate simulation results [11, 12].

In the domain of computer networking research, network emulation testbeds enable researchers to partially implement their experimental network designs. Network emulation is:

> a hybrid approach that combines real elements of a deployed networked application, such as end hosts and protocol implementations, with synthetic, simulated or abstracted elements, such as network links, intermediate nodes and background traffic [12].

Network emulation testbeds have numerous PCs interconnected through hubs or switches. These PCs, hubs and switches are actively configured by the testbed system to emulate nodes and links in experimental network design topologies. Once a network design is submitted, a mapping algorithm selects and configures available resources to build and execute the experiment for the desired duration. Nodes that make up the submitted topology are emulated by end-node PCs. Links are emulated by the combination of intermediate PCs, network switches and Category 5 (CAT5) network cables. Link characteristics and performance constraints, such as latency and packet loss, are modeled by intermediate PCs that delay or prevent portions of traffic from being passed from node to node [24]. In this way, testbed PCs represent a few select real hosts of the experimental network and can be programmed to execute developmental applications and protocols. Networks links and other hosts are synthetically represented by a blend of testbed resources [12].

Figure 2.7: Photograph of a portion of the 328 Emulab Classic rackmounted testbed computers at the University of Utah (January 2007).

2.2.3 University of Utah's Emulab. Emulab is a large network emulation testbed located at the University of Utah. Emulab consists of three independent testbeds that share their resources, meaning that components from all three can be combined in a single experiment [29]:

Mobile Wireless Laboratory. The nodes in this testbed are static and mobile wireless mote sensors. Portable motes are attached to remotely controlled robots that can be moved throughout the facility to replicate mobile sensor conditions.

Fixed 802.11 Wireless Laboratory. This testbed is made up of PC nodes that use 802.11 a, b and g wireless network interfaces. The testbed PCs are dispersed across various locations at the University of Utah testbed facility.

Emulab Classic. The original University of Utah testbed consisted of 168 rackmounted PCs of various hardware configurations. The latest version incorporates 328 PC nodes to handle the demands of additional users and larger virtual topologies. The PCs in this "cluster" testbed are connected to each other directly with CAT5, or through one or multiple network switches.

AFIT's CyberOpeRations Emulator (CORE) hardware layout is similar to the Emulab Classic testbed. Thirty-five PC nodes are rackmounted and connected to each other with hardwired Fast Ethernet network interfaces and a Cisco network switch [13]. AFIT and sixteen other universities use the same software testbed environment built and developed by the University of Utah for Emulab.

2.2.3.1 Mapping Algorithm Goals. A user "experiment" is the primary unit of workload for Emulab. Every action taken by Emulab supports user experiments, whether by actively running an experiment, syntax checking user scripts, mapping resources to virtual topologies, or monitoring experiment performance. A researcher builds an experiment script using a Java™ GUI accessible on Emulab's homepage or using the NS2 program [13, 29]. The script is parsed to ensure there are no errors and loaded into a database to await resource assignment. This database also serves as a repository to "swap in" idle experiments that no longer have resources assigned to them. If a pnode or other hardware fails, resource assignments in this database are used to expedite recreating the experiment. Thus only vnodes assigned to faulty hardware must be re-evaluated. Once the experiment is submitted, Emulab's software environment "maps" available testbed resources to the researcher's virtual topology, using a solver known as *assign*. *Assign* has five goals when it maps testbed resources to a virtual topology [14, 24]:

1. Correctly assign vnodes and vlinks to available pnodes and plinks by ensuring specified hardware, software and protocol configurations are met and no artifacts are introduced into the physical topology.
2. Map vlinks to plinks in such a way that inter-switch bandwidth in the physical topology is minimized.
3. Complete the mapping in such a way to maximize the number of experiments that can be run simultaneously on the testbed.

4. Facilitate experiment scaling by minimizing the number of pnodes required for each experiment. This is done by assigning multiple similarly-configured vnodes to a single pnode.

5. Complete the assignment process in a minimal amount of time, much lower than it takes a user to develop a virtual topology to expedite experiment creation time.

2.2.3.2 Node Types. A pnode in an experiment can be a fully interactive PC end-node, a router, a delay node to traffic-shape a link, or a host for multiple simulated vnodes [12]. A type system in the Emulab environment determines whether or not a pnode is a potential match for a vnode. Every vnode is given a type and every pnode is given a list of vnode types it can support. Pnodes are also given the number of vnodes they can simultaneously support for each type. Only vnodes of the same type can be mapped to the same pnode. For example, Figure 2.8 shows how any of the four pnodes can support vnodes `delay1`, `delay2` and `node1`. Vnode `node1` in Figure 2.8 (a) is of type `pc`. Vnodes `delay1` and `delay2` are both of type `delay`. All four pnodes in Figure 2.8 (b) are capable of hosting vnode types `pc` and `delay`. Additionally, vnodes `delay1` and `delay2` can be placed on the same pnode for a more efficient mapping. Only pnodes `pc1` and `pc2` are candidates for vnode `node2`, however, due to the request for an 850Mhz processor. A complete description of vnode, pnode, vlink and plink syntax in .top and .ptop files is provided in Appendix B [24].

2.2.3.3 Link Types. The Emulab environment supports four types of plinks that can be mapped to vlinks: intra-node links, direct links, intra-switch links and inter-switch links. Intra-node links are links between vnodes mapped to the same pnode. Intra-node links are physical from the perspective that they consume pnode memory resources when in use, but do not require additional hardware. Direct links are links between two pnode network interfaces that do not pass through a network switch. Intra-switch links are links between two pnodes that cross only one network

```
node node1   pc
node node2   pc850
node delay1  delay
node delay2  delay
```

(a)

```
node pc1 pc:1 pc850:1 delay:2
node pc2 pc:1 pc850:1 delay:2
node pc3 pc:1 pc600:1 delay:2
node pc4 pc:1 pc600:1 delay:2
```

(b)

Figure 2.8: (a) Example vnode descriptions from an Emulab .top file. Vnode descriptions are in the format node <node> <type> [<desires>], where <node> is the vnode string identifier and <type> is the string identifier for the vnode type. (b) Example pnode descriptions from an Emulab .ptop file. Pnode descriptions are in the format node <node> <type> [<desires>], where <node> is the pnode string identifier and <types> is a space-separated list of <type>:<number>. <type> is the string identifier for the vnode types this pnode can host and <number> is the number of vnodes of the particular type this pnode can host. A complete description of vnode, pnode, vlink and plink syntax in .top and .ptop files is provided in Appendix B [24].

switch. Inter-switch links are links between two pnodes that traverse more than one network switch [24].

Efficiently mapping vlinks to plinks supports *assign's* second and third goals, which are mutually inclusive. Restricting the use of limited testbed resources increases the probability of successfully mapping additional experiments while other experiments are running. Network switch nodes and inter-switch bandwidth are two examples of limited testbed resources. Even if an additional experiment successfully mapped its vnodes to pnodes, a poor vlink to plink mapping can oversubscribe inter-switch bandwidth. Oversubscription can create errors and artifacts in experiments that are not able to access the required amount of bandwidth [14, 24].

2.2.3.4 Virtual Equivalence Classes. Users are less concerned about selecting pnodes with newer hardware and more concerned that a set of vnodes are equivalent [24]. A virtual equivalence class (vclass) allows users to specify that all members of a vnode set be of the same type. For example, if a user wanted to create a vclass for a group of clients and another vclass for a group of servers. Vclasses are classified as hard or soft. Hard vclasses must be satisfied and violate fundamental constraints if broken. Breaking constraints for a hard vclass results in an infeasible configuration (discussed in Section 2.3.2.2). Soft vclasses can be broken by *assign*

in the search for a better configuration, but penalties are assessed by the objective function for doing so [24].

2.2.3.5 Features and Desires. Features are hardware and software resources offered by pnodes. Processor speed, hard disk space, Random Access Memory (RAM) size are examples of pnode hardware features. A pnode's preloaded operating system (OS) is an example of a software feature. Mapping a vnode to a pnode with the correct OS already loaded will eliminate the time required to install the requested OS. Desires are requests for features specified by the user. Good mapping solutions will mate features and desires to the greatest extent possible in user experiments. Unfulfilled desires and wasted features penalize solutions by increasing the cost of the mapping algorithm's objective function (see Section 2.3.2.1) [24].

Features and desires are weighted so that penalties are not equal when summed together in *assign's* objective function. Some resources are more limited than others, and scarcer resources carry a greater weight. For instance, it may be desirable for *assign* to choose a pnode that has a higher processor speed than required, over one with an extra Fast Ethernet connection. In this way, the penalty for wasted processing resources is less than that for an unused fast ethernet connection. In another case, wasting a fast ethernet connection may be preferable to choosing a pnode with a gigabit ethernet connection that is not specifically requested [24].

2.3 Metaheuristic Algorithms

Metaheuristics are a general class of algorithms for solving combinatorial optimization problems and problems that do not have an efficient domain-specific algorithm. The goal of combinatorial optimization is to maximize or minimize an objective or cost function to yield the best solution [35]. Throughout this thesis, the goal will be to minimize the objective function unless otherwise stated. Difficult optimization problems can be found in many fields, such as telecommunications, logistics, financial planning, transportation and mass production [10]. The network testbed mapping

problem is a difficult optimization problem and is known to be NP-hard when recasted as the traveling salesman problem [24]. Two widely used metaheuristics to solve difficult optimization problems are simulated annealing and tabu search [8].

Classic iterative or local search algorithms start from an initial configuration that is either chosen or randomly selected. The configuration is then modified in an attempt to improve the underlying objective function. Iterating from one configuration to another is known as a "move". The new resulting configuration is referred to as a "neighbor" solution. The objective functions of the initial and new configurations are compared, and the new solution is accepted if an improvement is achieved. Otherwise, the new solution is discarded and the search algorithm returns to the previous configuration. The algorithm terminates when attempts to modify the configuration fail to improve the objective function [8, 10].

Classic iterative algorithms often get "stuck" in local minima. Figure 2.9 illustrates this condition where c_0 represents the initial configuration. A local search algorithm will accept the next configuration c_1 as this configuration lowers the score of the objective function. The local search algorithm will continue to accept all configurations $c_2 \ldots c_n$ as these configurations also continue to lower the objective function. However, c'_n will be rejected as this configuration increases the cost of the function. The local search will terminate, never finding a better solution. Most importantly, the algorithm will never explore the global optima c^* [8, 10].

Metaheuristics differ from local search algorithms in that they have a much better chance of locating the global optima in objective functions. Metaheuristics authorize increases (degradations) in the objective function to escape local minima and explore other "valleys". In this way, configuration c'_n can be accepted in Figure 2.9 and the algorithm has the potential to locate c^*, the global optima of the objective function. Mechanisms are put in place to counter authorized increases and ensure the search algorithm does not diverge [8].

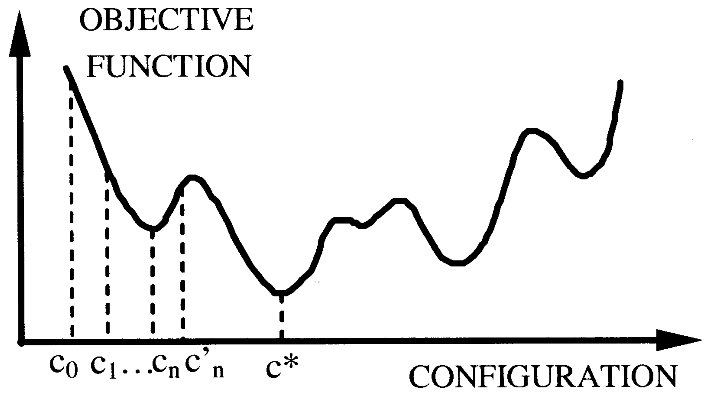


Figure 2.9: Landscape of an objective function of a difficult optimization problem. Different configurations are represented by c designators. The desired configuration is c^* that minimizes the objective function. [8].

2.3.1 Simulated Annealing. Simulated Annealing (SA) is a metaheuristic that mimics *annealing*, a slow cooling technique used by metallurgists to reduce defects and produce high-quality materials. In SA, a random initial configuration is chosen and neighbor configurations are selected at random to improve the objective function. A temperature parameter is used to determine whether or not increases in the objective function should be accepted. At high temperatures, nearly all configurations are accepted, allowing the algorithm to traverse "uphill" configurations and break out of local minima. As the temperature lowers, tighter restrictions are placed on the set of allowable configurations, resulting in fewer accepted configurations until the algorithm converges onto a final solution. SA is known for its flexibility and adaptability to a wide range of difficult optimizations problems. Management of the temperature cooling schedule can be difficult, but is crucial for a successful implementation [8, 24].

2.3.2 Assign - Emulab's Solver to the Testbed Mapping Problem. The *network testbed mapping problem* is defined as "the problem of selecting hardware to instantiate network experiments" [24]. The Emulab architects chose SA as the search algorithm because of its adaptability to a wide range of optimization problems. Unlike a typical SA search algorithm that starts with a random configuration, *assign* starts its SA algorithm with an empty one. *Assign* creates new configurations by changing

vnode assignments one at a time. Unassigned vnodes are given priority and mapped first. Once all vnodes are assigned, a randomly chosen vnode is remapped to create a new configuration. The objective function is used to score each new configuration. Violations, a concept unique to *assign*, are also summed for each new configuration. The configuration with the lowest score and lowest number of violations is retained as the best solution [24, 30].

2.3.2.1 Objective Function. *Assign's* objective function gauges the quality of each configuration. Configurations are scored according to the number and types of pnodes and plinks used in the physical topology. Scoring is not a trivial function, due to the complexity of features, desires, many-to-one relationships (mapping multiple vnodes to a pnode) and one-to-many relationships (a single vlink can span many plinks). A scoring system tallies the cost of wasted features, unfulfilled desires, soft vclass penalties and links for each vnode to pnode assignment based on the objective function. A cost for the number of pclasses used is also included. *Pclasses* are discussed in Section 2.3.3.1. An unfulfilled desire with a score greater than one results in a violation.

Table 2.1 shows the pnodes and plinks scores used in the objective function. Intra-node links are used first if possible, since their cost is the lowest. Inter-switch links have a cost much higher than the other links since they are one of the key resources to be conserved. Although Table 2.1 shows no penalty for the use of an intra-node link, this has been updated [11] since large virtual topologies can reach the upper limit on the number and capacity of vlinks that can be supported in pnodes. Scoring an entire configuration is not trivial and its complexity is $O(n + l)$, where n is the number of nodes in the configuration and l is the number of links. The scoring function needs to be computed quickly, due to the large number of times it is conducted in a single mapping. Updating the score incrementally each time a node is added, removed or reassigned lowers the computation time considerably to $O(l_n)$, where l_n is the number of links of the modified node [24, 30].

Table 2.1: Types of physical resources available in Emulab along with their their cost to *assign's* objective function [24].

Physical Resource	Cost
Intra-node Link	0.00
Direct Link	0.01
Intra-switch Link	0.02
Inter-switch Link	0.20
Physical Node	0.20
Switch	0.50
pclass	0.50

2.3.2.2 Violations. A high temperature means *assign* can consider configurations of lesser quality to escape local minima. Although these configurations are of lesser quality, they still represent valid physical topology solutions that fulfill user desires and constraints. A violation, on the other hand, are user desires or constraints that are not fulfilled by a given configuration. These configurations are considered "infeasible." Violations include, but are not limited to, oversubscribing inter-switch bandwidth, unfulfilled user desires, hard vclass penalties and unassigned vnodes. Two types of infeasible configurations are ones that lead to valid configurations and ones that do not. An example of an infeasible configuration that does not lead to valid solution space includes assigning a vnode with four vlinks to a pnode that supports only three vlinks. No matter how the rest of the virtual topology is remapped, this assignment always leads to an invalid solution. Exploration of these infeasible configurations is wasteful and inefficient. To prevent this, *assign* creates a list of valid pnode assignments for each vnode a priori [24].

Violations allow *assign* to consider infeasible configurations that lead to valid solution space as an additional method to escape local minima. Figure 2.10 shows an example of an infeasible configuration that leads to a lower minima. The left pane shows a locally optimum configuration. Pnodes are represented by circles. The upper left box shows a single pnode connected to a switch. The lower left box depicts a group of three pnodes all connected to each other via another network switch. Pnodes

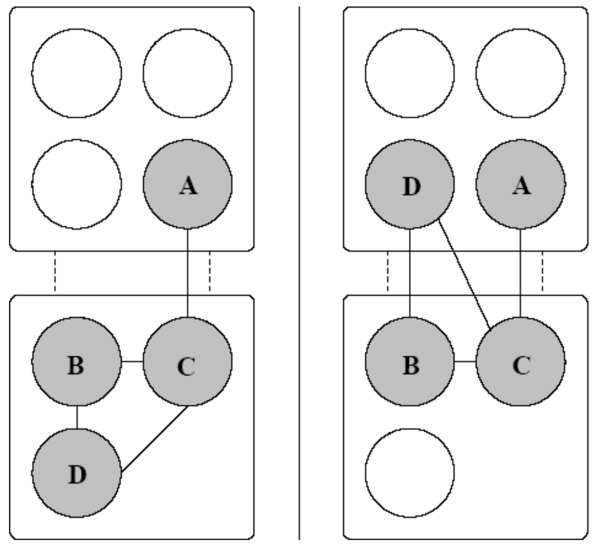


Figure 2.10: A situation in which traversing non-solution space to allows the configuration to migrate to a lower minima. Pnodes are represented by circles and vnodes are represented by capital letters. A mapped pnode is a grayed circle labeled with the vnode identifier. Pnodes that are connected to same switch are grouped together in a square. A pnode that communicates from one square to a pnode in another square must use an inter-switch link [24].

B, C and D communicate with pnode A through an inter-switch link. If the inter-switch link is saturated and cannot accommodate additional vlinks, the right pane represents an infeasible solution. However, the right-pane is a required intermediate configuration to reach a lower minima, in which all four pnodes reside in the upper box and communicate to each other with only intra-switch links [24, 30].

2.3.3 Improvements to Assign. *Assign* has been in use since January 2000 and has undergone many refinements to improve mapping performance. Physical equivalence classes and virtual graph coarsening have reduced the search space *assign* needs to explore, decreasing runtime considerably [14, 24]. Feedback-directed auto-adaptation of simulated resources alerts the Emulab environment that a pnode in the testbed network is overloaded with too many vnodes and the physical topology may no longer be valid [11].

2.3.3.1 Physical Equivalence Classes. Testbed facilities will typically have large sets of nodes with identical hardware. Remapping vnodes to a pnode with identical hardware often results in a configuration with the same objective function score and the same number of violations. Grouping pnodes together in a physical equivalence class (pclass) prevents *assign* from exploring identical configurations and dramatically reduces the search space. For a set of pnodes to be equivalent, they must have "identical types and features" and there must exist "a bijection from the links of one node to the links of the other that preserves destination and bandwidth" [24]. Without pclasses, the branching factor of *assign* is $O(v \cdot p)$, where v is the number of nodes in the virtual topology and p is the number of PCs in the testbed. In the 2003 version of the Emulab testbed, p was reduced from 168 testbed PCs to only 4 pclasses. When pclasses are enabled, *assign* chooses a pclass for assignment to a vnode, rather than a single pnode.

The effect pclasses have on reducing the search space breaks down when vnodes are multiplexed onto pnodes. A pnode with one or more vnodes is no longer equivalent to the rest of the empty pnodes in its pclass. Therefore, it must be removed to form its own pclass. *Assign* attempts to mitigate this by dynamically computing pclasses during the mapping process. Unfortunately, results thus far have been "close to not having physical equivalence classes at all" [11]. This is problematic because complex virtual networks leverage vnode multiplexing so as not to monopolize testbed resources. It is these large networks that stand to gain the greatest runtime reduction offered by pclasses [11, 24].

2.3.3.2 Coarsening the Virtual Graph. The next attempt to reduce *assign's* runtime focused on the virtual, as opposed to the physical topology. A key observation is that in good solutions, two adjacent vnodes have a high probability of being mapped to the same pnode when vnode multiplexing is used [14]. The goal, then, is to find vnodes in the user's virtual topology that map to the same pnode. This is accomplished by executing two algorithms prior to running *assign* on

the virtual topology. The first algorithm combines leaf nodes belonging to a Local Area Network (LAN) or similar network cluster into a single composite vnode. In a SF network, this is akin to combining a hub node and all connected nodes into one vnode. The second algorithm uses the METIS [17] graph partitioner to further combine vnodes produced by the first algorithm. METIS partitions the revised virtual topology in such a way that the average partition will fit in the pnode (pclass) with the least amount of resources. These partitions are then combined once more to form vnodes, producing the final virtual topology that is fed into *assign*. The attributes of the vnode "conglomerates" are a summary of the properties of the original vnodes. For instance, the memory requirements of a vnode conglomerate is a total of all the memory requirements of the original vnodes [14].

METIS is used because it has a much lower execution time than *assign*. It is faster because it ignores the complexity of matching vnode desires to pnode features. This can produce an outcome known as "fragmentation" in which the preprocessing algorithms create conglomerate vnodes that do not pack into pnodes as efficiently as they would if the virtual topology was mapped directly by *assign*. It can also create a second situation where no pnodes can handle the resource demands of the conglomerate vnodes. By carefully tuning the target size of the vnode conglomerate, [14] has lessened the impact of fragmentation. The worst fragmentation noted only increased testbed resources usage by 13 percent, an acceptable tradeoff considering a factor of 14 speedup in mapping 100 nodes and a factor of 28 when mapping 1000 nodes. The technique of automatically adapting the vnode to pnode packing ratio based on feedback was developed to combat the overloaded pnode problem [11, 14].

2.3.3.3 Feedback-Directed Auto-Adaptation of Simulated Resources. Poor mapping can result in simulated nodes not being able to keep up when communicating with real nodes. Often this is due to an "overloaded" pnode that does not have enough resources to meet the demands of all the vnodes mapped to it. Automatically optimizing the number of vnodes that can fit into a pnode accomplishes

two goals. The first is finding the best balance between efficiently mapping as many experiments to the testbed as possible and ensuring all vnodes, simulated and real, can keep up with each other in real time. The second is keeping user intervention to a minimum, allowing researchers to focus on their experiments instead of the Emulab platform running those experiments [11, 14].

A baseline must first be established that determines how many vnodes in a particular experiment can fit into testbed pnodes. This is accomplished by having the user execute a manual run, if the virtual topology is small. If the virtual topology is large and complex, a feedback-directed adaptation routine automatically determines the optimum packing factor. If a pnode overload is detected during an experiment, the the faulty pnode is remapped with a more conservative vnode packing factor based on feedback data collected [11, 14].

2.3.4 Tabu Search. Tabu Search (TS) is a metaheuristic first introduced by Fred Glover in 1986. TS is a local search algorithm, similar to SA, that iterates from one neighbor configuration to another until the stopping criteria is reached. In contrast to SA, TS mimics the concept of memory, as opposed to annealing, to solve difficult optimization problems. Memory guides TS towards good solutions based on information collected during the search. Memory is applied in TS in two parts, short-term and long-term. Short-term memory is implemented using a tabu list. Long-term memory can be implemented in many ways. Two popular methods are preventing the exploration of the same configuration repeatedly and forcing the visitation of solution space that has not been explored in a long period of time [8, 10].

Original incarnations of TS committed entire solutions to memory. For problems with large configurations, such as *assign's* mapping of complex networks, storing thousands of sets of virtual topology assignments causes significant growth in the mapping computer's RAM requirements. Additionally, it is time consuming to parse quickly through the data structure containing these configurations. Even using hashing tables to reduce storage demands and quickly locate configurations in RAM, space

requirements can still be high. Later versions of TS streamlined memory needs by storing only the moves that led to given configurations in memory, rather than the entire configurations themselves [8, 10].

2.3.4.1 Candidate List. SA uses random selection to iterate from one configuration to the next. In this respect, SA only considers only one move, the random selection, to create the next new configuration. TS attempts to intelligently select the next move by first evaluating numerous neighborhood configurations. The one most likely to improve the objective function that is not tabu (explained in Section 2.3.4.2) is selected. In this way, the search is directed towards high quality solution space. The structure used to rank order future moves is known as the candidate list. To accelerate the time it takes to locate a potential move, only a subset of all possible future moves are included in the candidate list [8].

2.3.4.2 Tabu List. The tabu list is the chief component of TS. The tabu list contains a list of moves that, as a result of recent past moves, should not be chosen. The purpose is to prevent recent configurations from being revisited. If a move just selected changes to a new configuration, the reverse of that move should not be permitted because the configuration will return to its original state. This creates a situation in which very few unique solutions are found because the search has explored only a small subset of possible configurations. To prevent this, the next time a move is chosen from the candidate list, it is first compared against the tabu list to see if it is allowed [8, 10].

2.3.4.3 Tabu List Length. The tabu list length corresponds to the number of forbidden moves for the current iteration. Referring to the objective function landscape shown in Figure 2.9, the greater the number of tabu moves (longer list length), the more likely the search will escape the local minima valleys. This is known as *diversification*. Diversification stimulates the search to visit new regions of solution space and creates configurations that vary greatly from each other. Too

much diversification can cause the search to miss better local minima, possibly even the global optimum, by skipping completely over valleys. *Intensification* is a way to reverse this effect. Intensification occurs when the list length is short, enabling the search to more thoroughly probe the current valley. Intensification exploits the fact that high quality solutions often share the same attributes and is used to locate the best configuration in a region of good solution space [8, 10].

The list length should not be too long, otherwise all possible moves may be excluded. Leaving only one or two legal moves per iteration may also not be desired, since the search is more heavily influenced by the few available moves rather than the objective function. *Aspiration* is the allowance of a tabu move if it improves the objective function. Aspiration can be used to counter the impact of a long list length. It is important to point out that aspiration should never be used to compel a certain move to be taken, rather simply make the tabu move available. The objective function should be used to determine whether or not the move is chosen. In addition to being too long, the list length should also not be too short or cycles may appear. Cycles occur when recent solutions are constantly revisited, the event the tabu list is designed to prevent [8, 10].

The list length is a difficult parameter to set, but crucial to the good performance of TS. A static list length will result in poor performance due to the fact that the optimal length is closely tied to characteristics of the current problem. It can become cumbersome to re-evaluate the best list length for each new problem instance. Varying the list length throughout the search alleviates this problem by intensifying or diversifying the focus for certain amounts of time. Two popular methods of varying the tabu list are randomly and reactively. Reactive TS bases the list length on feedback gained during the search [5]. Robust TS randomly varies the list length between a minimum and maximum length parameter [8].

2.3.4.4 Long-term Memory. Diversification is often not enough to ensure TS will explore all regions of the solution space. If some regions are left

unvisited, then the global optimum can be missed. Long-term memory is used to coerce the search into solution space that has not been investigated. One way of pushing the search into new solution space is by preventing the same movement from taking place repeatedly. Long-term memory facilitates this by maintaining statistics on moves taken throughout the search. Moves often selected are penalized, typically with a weight proportional to their frequency. Another method takes the opposite approach, by forcing a move that has not been used in a long period of time, regardless of the impact to the objective function. This approach forcefully vaults the search out of the valley it was exploring into another region of solution space [8].

2.4 Summary

This chapter provides background information on complex networks, the network performance evaluation technique of emulation, SA and TS metaheuristics. Chapter 3 describes the research methodology to compare the performance of *assign* using a SA search algorithm with a search algorithm based on TS.

III. Research Methodology

This chapter outlines the research methodology to compare the performance of *assign* using a SA search algorithm with a search algorithm based on TS. A systematic approach is used to analyze the performance of both search algorithms. The problem definition and experimental goals are clearly defined following this introduction. Rationale and description of the evaluation technique, factors selected, performance metrics, experimental design and workload is also provided. Statistical analysis of the data collected is presented in Chapter 4.

3.1 Problem Definition

The principal aim of Emulab is to provide a network emulation testbed for researchers. This objective has many competing constraints that must be realized. A useful network emulation testbed would rapidly create and deploy high fidelity, reliable experiments that produce trustworthy results. Another concern is the transparency of the testbed with regards to the OS and applications under test. The testbed management system must not interfere with a running experiment to better recreate real-world situations. Scalability permits experiment of complex networks and other large virtual topologies [14]. Simulation has achieved widespread acceptance in academic and research communities by enabling users to rapidly deploy and analyze experimental networks [20]. Emulab strives to equal the ease-of-use and rapid experiment deployment that has made simulation popular, coupled with the realism offered by emulation [29].

There are a number of requirements that must be met to rapidly deploy virtual topologies onto an Emulab testbed. The time required for a user to navigate the Emulab webpage GUI and create a virtual topology, how fast *assign* can find a worthy solution, and how long it takes testbed PCs to load custom OS images are just a few of the considerations that must be taken into account to quickly construct representative physical topologies. Developing quality solutions for complex virtual topologies with thousands of nodes in a minimum amount of time remains a challenge for *assign*.

Many improvements have been made thus far, as the original Emulab could only instantiate 100-node experiments while the current version reliably maps experiments up to 2,000 nodes in approximately three to four minutes [14, 24].

This research is concerned with the problem of creating high quality, feasible solutions for complex networks with thousands of nodes in a minimum amount of time. 'Quality' refers to the score of the objective function, a lower score indicating a more optimal solution. Higher quality solutions make more efficient use of testbed resources allowing scalability of experiments and more simultaneous users. 'Quality' also refers to violation count, as feasible physical topology solutions have zero violations. 'Minimum' refers to the amount of time required by *assign* to locate a feasible solution for a virtual topology. The time required to locate a feasible solution should be minimized to the greatest extent possible. The time required to instantiate a user experiment on an Emulab testbed is of greater importance than objective function score. A mapping time on the order of hours to find the lowest possible objective function score is unacceptable since availability of testbed resources can change within minutes. *Assign* would then be forced to restart until a solution was reached where all testbed resources were available [24]. The worst case mapping time should be much smaller than the time it takes for a PC node to reboot. Other than loading custom disk images, node rebooting dominates experiment creation time [33].

3.1.1 Goals and Hypothesis. Metaheuristic algorithm runtime can be reduced by reducing the search space or branching factor. The branching factor in *assign* is $O(v \cdot p)$, where v is the number of nodes in the virtual topology and p is the number of PC nodes in the testbed. Coarsening the virtual topology using a graph partitioner such as METIS effectively reduces the value of v [14]. Pclasses are a useful method to reduce p, except in large topologies that require vnode multiplexing. In these cases, a partially filled pnode is no longer equivalent to the rest of the pnodes in the pclass. A partially filled pnode must therefore be removed to form its

own pclass. This occurs throughout the mapping process mimicking the effect of not having pclasses at all [11].

Another way to reduce the runtime of a metaheuristic algorithm is to improve the search technique. SA relies on a large number of iterations to produce a good quality solution, as the algorithm is guided almost exclusively by chance. Poor quality solutions must be traversed before a smaller subset of good solutions is found. TS incorporates the use of memory more quickly to direct the search towards higher quality solution spaces [8].

The goal of this research is to determine whether a TS implementation of assign *is superior to Emulab's existing SA implementation with respect to execution time and solution quality.* The application of short-term memory prevents TS from revisiting solutions in the same manner as SA. Long-term memory ensures TS will explore a larger region of solution space than SA, giving TS a better opportunity to locate lower minima. It is expected that a TS implementation of *assign* will locate physical topology solutions in less time than Emulab's existing version of *assign* when mapping identical virtual topologies. It is further expected that the number of violations and the objective score of TS solutions will be equal to or lower than SA solutions for the same virtual topologies.

3.1.2 Approach. To achieve the research goal, the violation count, objective score and execution time produced by the original version of *assign* is compared with a version modified to use the TS algorithm. A workload composed of 38 virtual topologies and one set of available testbed resources is submitted to the original SA and TS versions of *assign*. Each version of *assign* produces a solution consisting of a physical topology from the provided virtual topology and one of the sets of testbed resources. Both versions of *assign* are implemented in the C programming language and compiled using the test environment specified in Table 3.3 and the makefile listed in Appendix A. *Assign* is launched from the command line in a console window within the K Desktop Environment, on a Dell laptop computer running the FreeBSD

OS. The virtual topologies are specified in a text file with a .top extension. Similarly, the available testbed resources are denoted in a text file with a .ptop extension. The command line arguments are the .top file, the .ptop file and runtime options shown in Table 3.4. The .top and .ptop files used in testing are listed in Appendix B. Upon completion, *assign* outputs the physical topology, violation count, objective function score and execution time to the console window. Objective function score results from the SA and TS algorithms are compared for each of the virtual topologies to determine which was lower, indicating a higher quality solution. Runtimes for both algorithms are also recorded to determine which had a lower execution time. If violations are present in a physical topology, *assign* is considered to be unable to reach a valid solution for the given virtual topology. Objective function score and execution results are not compared for these cases.

The range of complex networks under study is constrained to 38 SF and random networks ranging from 10 to 1000 nodes. These complex networks make up the virtual topologies submitted to *assign*. The output of *assign* would normally feed back into the Emulab system to reserve selected physical resources. Allocated physical resources such as PCs would then boot up and load custom images in preparation for the user's experiment. For the purposes of this research, the physical topology solution is sent to the computer display at the end of the search using diagnostics built into *assign*.

3.2 System Boundaries

Emulab is a system of systems. Figure 3.1 shows the overall Emulab system architecture. The User Interface is the portion of the Emulab system that a user directly interacts with to build a virtual topology either through the webpage GUI or NS2 script. Accounts and Database house member login accounts and privilege levels. The MySQL® database stores the virtual topology specification after being parsed from the webpage GUI or NS2 script. The MySQL® database holds idle experiments state so testbed resources can be released to other experiments. Idle experiments can be reestablished much faster from the MySQL® database than from

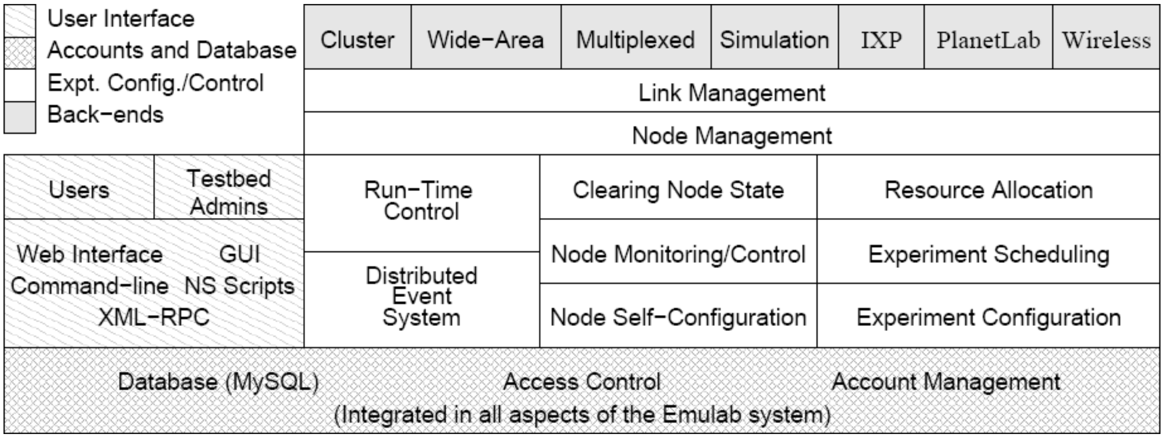


Figure 3.1: The various components of the Emulab system architecture [12].

the original instantiation since vnode assignments are retained and only resources in conflict must be remapped. The Experiment Configuration and Control segment has resource assignment mechanisms (e.g., *assign*) and systems to configure, monitor and control experiments in progress. The back-end is the physical testbed elements including those described in Section 2.2.3. Resources from any one or all testbeds can be selected and combined into a user experiment [12].

The System Under Test (SUT) is the testbed mapping algorithm known as *assign* and shown in Figure 3.2. *Assign* is a subsystem in the overall Emulab system located in the Resource Allocation block in the Experimental Configuration and Control segment. *Assign* includes the search algorithm, the objective function the search algorithm is attempting to optimize, the scoring and violation system that gauges solution quality, the subroutine that generates pclasses, node addition and removal processes, the link resolution process, and mapping and type prechecks.

The objective function and scoring system is discussed in detail in Section 2.3.2.1. The objective function determines solution quality based on score. The scoring system tallies a configuration's score by summing the penalties for each pnode and plink assignment. Pclasses are generated immediately prior to mapping since the list of available testbed resources is constantly fluctuating. The search algorithm interfaces with the scoring system through node addition and removal processes. As the search

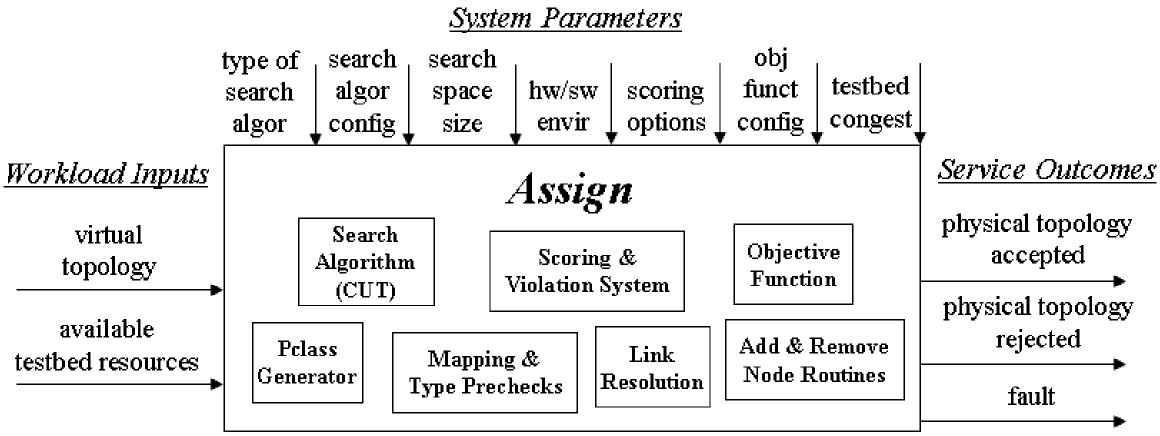


Figure 3.2: The system boundaries of the testbed mapping algorithm.

algorithm selects a vnode to be added, removed or both (in the case of reassignment), the node addition and removal processes trigger the scoring system to modify the score and violation count. Hence the score is updated incrementally as opposed to all at once. The link resolution process is much more streamlined than the node mapping process, as *assign* simply finds all possible links between connected nodes and chooses one. Plinks are chosen according to their cost (e.g., intra-node and direct links are chosen before intra- and inter-switch links). A mapping precheck creates a general list of pnodes that are acceptable mappings for each vnode, preventing *assign* from exploring infeasible configurations that do not lead to valid solutions. A type precheck ensures there are available pnode types to match user-specified vnodes and vclasses.

The Component Under Test (CUT) is the search algorithm, either SA or TS. *Assign's* original SA algorithm is used as the baseline against which the TS version is measured. Only virtual topologies that meet SF and random criteria are considered. The problem of differentiating between networks that have random, small-world, SF or other characteristics is beyond the scope of this research. The virtual topologies are not "coarsened" in advance to reduce the number of vnodes and vlinks by means of METIS or another graph partitioner. Graph coarsening takes place outside of *assign*, and is not part of the SUT. Only "cluster" resources from a testbed such as

Emulab Classic or the AFIT CORE make up the set of testbed resources. Testbed resources are considered available until the current mapping algorithm selects them for assignment. Changes in resource availability due to testbed congestion or equipment failure is not within the scope of this research.

3.3 System Services

Assign creates a proposed physical topology by mapping a user-submitted virtual topology to available testbed resources. A successful outcome occurs when a physical topology solution with no violations (a feasible configuration) is produced. This physical topology is a specification that can be immediately instantiated on available testbed resources and is thus "accepted" by the Emulab system. A failure happens when either the physical topology is rejected or a fault occurs while *assign* is running. A physical topology might be rejected because required testbed resources are unavailable due to slow mapping time (e.g., another mapping process reserved a resource that was initially available) or equipment failure (e.g., an initially available resource is no longer available due to hardware or other problems). A fault arises when the virtual topology is flawed (e.g., syntax errors or requests for resources that do not exist) or due to a condition that causes *assign* to terminate before arriving at a physical topology solution. A logic error within the *assign* program or the failure for the submitted virtual topology to pass mapping or type prechecks are examples of such conditions. A fault can also occur because no physical topology solution exists for the virtual topology with the testbed resources currently available.

Only successful service outcomes are analyzed. The focus of this research is to compare mapping algorithm performance, therefore, rejected physical topologies due to testbed resource unavailability are not considered. Both versions of *assign* are checked by hand, using the 10-node SF virtual topology, to ensure that feasible physical topologies are produced and no logic or other program errors exist. Pilot tests are conducted to ensure all virtual topologies pass mapping and type prechecks. All virtual topologies that make up the experiment workload are syntactically correct

and are checked to ensure no other flaws exist. A physical topology solution exists for each virtual topology with the available testbed resources provided in the .ptop file.

3.4 Workload

The workload of the system is the virtual topologies in the form of .top text files and the set of available testbed resources described in the .ptop text file. The workload is artificially created. The .ptop file has 525 PC nodes, more than any known network emulation testbed in existence. The number of ethernet plinks per pnode in the .ptop file ranges from 4 to 20. Most Emulab testbeds incorporate PC nodes that have two to six plinks. Since SF networks have a large number nodes, and some of these nodes have a large number of links, the synthetic testbed described in the .ptop file allows a greater range of feasible physical topologies than current existing Emulab testbeds. Specifically, it allows a greater number of vnode-pnode and vlink-plink one-to-one mappings. The availability of low-quality assignments helps determine which search algorithm more tightly "packs" experiments onto testbed resources.

A visual depiction of the .ptop file is shown in Figure 3.3. The synthetic testbed is comprised of eight network switches. Seven PC network switches interconnect the testbed PC nodes. One master network switch connects the seven PC network switches together. Each of the PC network switches is connected to 75 PC testbed nodes. Each PC node is connected to its respective PC network switch by multiple gigabit ethernet links. All PC network switches connect to the master network switch by a single ten-gigabit ethernet link. Table 3.1 lists the number of gigabit plinks per PC node and the PC network switch to which each node is connected. Few of the PC nodes have greater than five gigabit plinks. This is representative of actual Emulab testbeds as older generation workstations with a small number of network interfaces represent the majority of testbed PC nodes. PC nodes with a large number of network interfaces are newer, much more expensive to purchase and maintain, and are therefore present in fewer quantities in most testbeds.

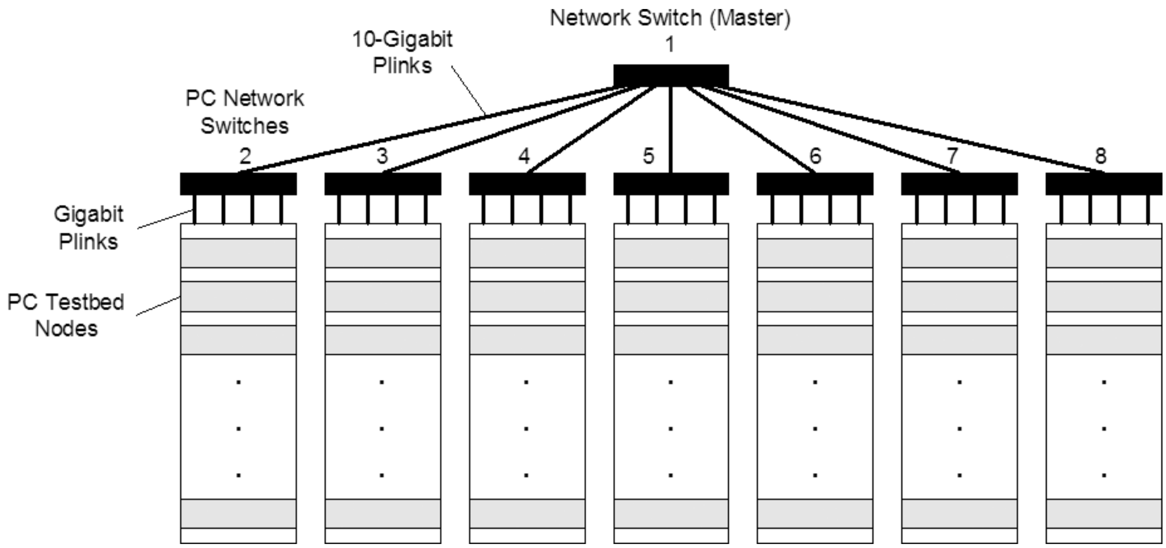


Figure 3.3: A diagram of the synthetic testbed described by the .ptop text file. The eight black boxes represent network switches. The PC testbed nodes are indicated by gray boxes. Only four of the 75 PC nodes are included in each rack (larger white box) for notional purposes. All of the 75 PC nodes in each rack are connected to the PC network switch displayed at the top of the rack by multiple gigabit ethernet plinks specified in Table 3.1. Each of the seven PC network switches are connected to the master network switch by a single ten-gigabit ethernet plink.

The virtual topologies consist of 38 SF and random networks ranging from 10 to 1000 nodes. The .top files are also synthetically generated, but represent workloads a researcher would submit for study, as the virtual topologies are proportional in size and complexity. Virtual topologies are created using the Boston university Representative Internet Topology gEnerator (BRITE) [19] based on the BA SF and ER random graph models. The BRITE SF model incorporates both concepts key to BA SF networks, incremental growth and preferential connection. The BRITE SF networks start with a single node and as each additional node is added to the topology, links to existing nodes are established based on preferential attachment. Figure 3.4 shows the 500-node BRITE SF virtual topology using Otter [15], a general purpose network visualization tool. The three main hubs and self-similar SF characteristics can easily be seen in Figure 3.4. Small-world networks are not tested. SF networks are most structured of the three types of complex networks and are assumed to represent the worst-case workload. Conversely, random networks have the least structure and

Table 3.1: The number of gigabit ethernet plinks each PC node has connecting to its corresponding PC network switch. For PC node categories that do not all connect to the same switch, the network switch number is shown first followed by the amount of PC nodes connected to it in parenthesis.

PC Node	**Number of Plinks Per PC Node**	**Network Switch**
1-5	20	2
6-20	15	2
21-105	10	2(55), 3(30)
106-170	5	3(45), 4(20)
171-525	4	4(55), 5(75), 6(75), 7(75), 8(75)

represent the majority of virtual topologies submitted by Emulab users. The .top and .ptop files used in testing are described in Appendix B.

3.5 Performance Metrics

The performance metrics used in this study are objective function score, violation count and execution time. *Assign's* objective function score is a summation of pnode and plink penalties. Violations determine whether a configuration is feasible or infeasible. The objective score and violation count together measure the quality of a physical topology solution. The solution quality is an indicator of how well user desires are satisfied and how many testbed resources are required to instantiate the experiment (i.e., how well the experiment is "packed" onto testbed resources). The execution time is the number of seconds *assign* requires to find a feasible solution to the virtual topology specified in the .top file.

The primary performance metric for this study is whether a feasible physical topology solution is produced from the provided virtual topology and the set of available testbed resources (i.e., a successful service outcome). SA and TS are sub-optimal search algorithms, hence a feasible physical topology solution is not guaranteed every time the search algorithms have completed. The goal of this research is to determine

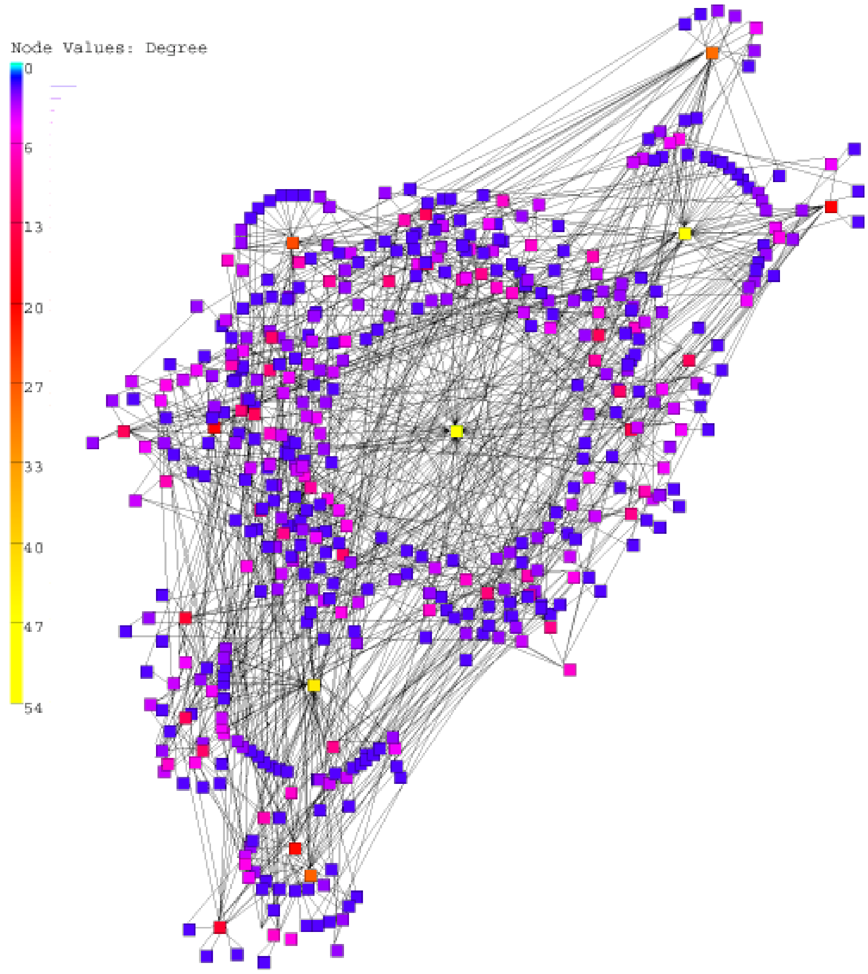


Figure 3.4: A graphic of the 500-node BRITE SF virtual topology with 997 links using Otter, a general purpose network visualization tool. The nodes are colored by their degree value. Three SF "hubs" created by incremental growth and preferential attachment are shown in yellow. Also apparent are the SF self-similar and fractal characteristics.

whether a TS implementation of *assign* is superior to Emulab's existing SA implementation in terms of execution time and solution quality. Thus, physical topologies produced by SA and TS that accurately represent submitted virtual topologies (i.e., feasible solutions) must be available for comparison. Execution time has a higher priority than objective function score. A testbed mapping algorithm that can quickly find a feasible solution to a user's virtual topology lowers the overall experiment creation time and increases interactive use of the testbed. This is preferable to increasing execution time in an attempt to reduce objective function score, lowering the amount

of interactivity provided by the Emulab testbed. The total number of iterations performed by the search algorithm and the number of iterations it took to reach the best solution are recorded, but not analyzed. Future research may use this additional information for other purposes, such as characterizing how termination conditions affect search algorithm performance.

3.6 Parameters

System and workload parameters that affect *assign's* mapping performance are shown in Figure 3.2 and described below. Most system parameters are *assign* runtime options set at compile time or at execution time via the command line. The remaining system parameters are environmental (e.g., hardware platform on which *assign* is running or amount of simultaneous testbed users). Workload parameters consist of only the virtual topology and available testbed resources.

3.6.1 System.

- The type of search algorithm is the primary system parameter. It determines how vnodes are assigned to pnodes in the physical topology.
- The configuration of the search algorithm includes algorithm-specific options, termination conditions and diagnostic subroutines. The tabu list length and SA cooling schedule are examples of algorithm-specific options. Algorithm-specific options determine how meticulously the search algorithm examines the search space. More careful inspections result in longer runtimes and may yield higher quality solutions. Coarser inspections complete in less time but may miss high quality solutions. Termination conditions may or may not be unique to the search algorithm. Halting the search after a specified number of iterations is an example of a general termination condition that is not unique to the search algorithm. Termination conditions dictate when the search is complete and prolong or shorten runtimes. Termination conditions may reduce solution quality if the search algorithm is stopped after too brief a time period. Diagnostic subroutines

track how the search is proceeding, but incur high overhead and typically slow down execution time due to large amounts of information processed during the search. An example of a diagnostic is displaying the current solution at every iteration. Diagnostics are normally disabled when the search algorithm is in use.

- The size of the search space greatly impacts search algorithm execution time. A large search space requires a longer time to examine than a small search space at the same level of detail. A smaller search space will not reduce solution quality if identical or infeasible solutions are removed.

- The hardware and software platform that hosts *assign* can affect search algorithm execution time. Processor speed, RAM size, compiler version and compiler optimizations can speedup or slow down execution time. Solution quality will remain constant on faster or slower hardware, as long as the search does not abnormally terminate due to lack of computing resources.

- Scoring and objective function options change both solution quality and execution time. Adding or removing violations and changing the weight of a penalty can cause the search algorithm to arrive at a sub-optimal solution faster or slower than previous settings.

- Testbed congestion can lower solution quality compared to a search where all testbed resources are available, if components required for a higher quality solution have been allocated by another user. Testbed congestion can also lower execution time since search space is reduced due to unavailability of testbed resources.

Table 3.2: *Assign* factors and levels for search algorithms.

Factors	Levels
Search algorithm	Simulated Annealing, Tabu Search
Network type	Scale-free, Random
Number of vnodes	10, 20 30, 40, 50, 60, 70, 80, 90, 100, 200 300, 400, 500, 600, 700, 800, 900, 1000

3.6.2 Workload.

- The virtual topology submitted by the user is the primary workload parameter. A large number of vnodes and vlinks in the virtual topology increases the search space examined by the search algorithm, inflating execution time.
- The number of available testbed resources can also increase or reduce search space. A larger number of testbed resources may broaden search space and increase execution time, but may also increase solution quality since there is a higher probability that available pnodes will have features that are a good match for vnode requirements.

3.7 Factors

Factors are parameters varied during experimentation [2]. Since the goal of this research is to determine which search algorithm is superior, the first factor is the search algorithm type. This research is concerned with complex networks, so the second and third factors are the network type and number of nodes in the submitted virtual topology. Table 3.2 lists the factors and levels in *assign*. Values for parameters not chosen as factors are listed in Table 3.3, Table 3.4 and Appendix A. It is anticipated that the TS algorithm will produce solutions of equal or higher quality than *assign's* original SA search algorithm. Execution times to arrive at these solutions are expected to be lower for the TS than the SA algorithm.

Table 3.3: The hardware and software test environment used to analyze the performance of *assign*.

Component	Value
Computer Make & Model	Dell Latitude D620 laptop computer
Processor Type	Intel® Core™ Duo processor
Processor Clock Speed	2.16 GHz
RAM Size	2048 MB
Operating System	FreeBSD version 6.1 release 0
Desktop Environment	K Desktop Environment release 3.5.1
Integrated Development Environment	KDevelop release 3.3.1
Assign Version	20061122
C and C++ Compiler	GCC version 3.4.4

3.8 Evaluation Technique

There are no analytical or simulation models to evaluate *assign*, therefore the evaluation technique is direct measurement. *Assign* is a real system that has been used by Emulab since January 2000. *Assign* is written in the C program language and can be compiled and executed on any computing platform able to host the Emulab system. The availability and portability characteristics of *assign* make direct measurement an appealing evaluation technique. The physical environment and environmental variables used for testing are listed in Table 3.3. The remaining portion of the experimental setup consisting of compiler optimization level, SA cooling schedule options and other makefile configuration settings are specified in Appendix A.

Assign is launched from the command line in a console window within the K Desktop Environment. The command line arguments are the virtual topology file (.top), the available testbed resources file (.ptop) and any runtime options. The .top and .ptop files used in testing are listed in Appendix B. Table 3.4 shows the runtime options used in testing. Virtual topologies are submitted one at a time. Each time a virtual topology is submitted, *assign* chooses a seed to initialize its random number generator. To reproduce the results in this research, preselected random seeds shown are submitted via the command line. Only one instance of *assign* is executing at any given time. There is no testbed congestion and all testbed resources remain

Table 3.4: The runtime options used to analyze the performance of *assign*.

Option	**Description**	**Value**
-s <seed>	Random Number Generator Seed	*varies*
-P	Prune Unsuable Pclasses	*n/a*
-H <float>	Branching Factor or Neighborhood Size Multiplier	1.0

available while *assign* is executing. The computer clock measures the time for each search algorithm to find a feasible solution. The execution time, objective function score and violation count is output to the console window when the search algorithm terminates. The objective function scores, number of violations and execution times for each virtual topology is shown in Appendix C.

The virtual topologies are submitted to *assign* from the smallest topology to the largest, in the order shown in Table 3.2. The smallest topology, consisting of ten nodes, is submitted first. This topology is used to validate the physical topology solutions produced by both versions of *assign*. Validation is accomplished by comparing the resultant physical topology solution produced by each version of *assign* to the original 10-node SF virtual topology.

3.9 Experimental Design

The experimental design is a full factorial experiment consisting of combinations of the factors and levels shown in Table 3.2. A full factorial is selected due to the small number of factors and levels and because it "tests every possible combination at all factor levels" [2]. Since there are three factors, two with two levels and one with 19, a full factorial design requires 76 experiments. Experiment repetition is based on variance of the results collected. Initially, 200 repetitions will be conducted. Additional repetitions will be conducted if required by the analysis. The resolution of the measurements is determined by the computer's system clock and the scoring system.

3.10 Implementation

The TS algorithm is implemented by modifying the anneal.cc source code. Anneal.cc is the C code implementation of the SA search algorithm in *assign*. Anneal.cc is a function called by assign.cc, *assign's* source code in the Emulab testbed software environment. Prior to launching the SA search algorithm, *assign* initializes its random number generator using a preselected seed, reads in the available testbed resources by parsing the .ptop file, calculates the shortest paths to all available switches using a minimum spanning tree algorithm, and reads in the virtual topology .top file. Pclasses are generated to reduce search space. Mapping and types prechecks are conducted to prevent exploration of unnecessary configurations and to further reduce search space.

3.10.1 Original Anneal.cc using SA Search Algorithm. The original anneal.cc consists of two loops, one loop embedded in the other. Prior to entering the outer loop, "fixed" vnodes are assigned to specified pnodes. Fixed vnodes are vnode assignments specifically requested by the user or vnodes that were successfully mapped by a prior instance of *assign*. Fixed vnodes are removed from the set of vnodes to be mapped, thus decreasing search space. As pnodes are allocated, empty pclasses are removed also decreasing search space. The branching factor $O(v \cdot p)$ represents an worst case or upper bound, as v, the number of vnodes, is reduced by graph coarsening and the removal of fixed vnodes. p, the number of pnodes, is lowered by the use of pclasses, mapping and types prechecks, and the removal of empty pclasses. After all fixed vnodes are assigned, remaining vnodes constitute the set of unassigned vnodes. An initial objective function score and number of violations is calculated based on the set of unassigned vnodes and their vlinks. This initial score is invariant and is compared to the score of the final physical topology after all vnodes are unassigned. If these two scores are not equal, *assign* notifies the user the final solution may be invalid.

Each outer loop iteration corresponds to a temperature step in the SA cooling schedule. The first iteration is the melting period. The goal of the melting period

is to determine an initial temperature such that every possible move and resulting configuration is accepted. The second and later iterations are chill periods. The goal of the chill periods are to lower the initial temperature until only a single configuration is accepted, which becomes the physical topology solution. The temperature is lowered based on the standard deviation of objective function scores of accepted configurations. The outer loop terminates when the derivative of the average temperature change is smaller than a specified epsilon value of 0.0001. After the outer loop is exited, the current configuration reverts to the best known configuration, and the SA search algorithm returns this configuration to *assign* as the final physical topology solution.

Each inner loop iteration creates a new configuration and compares its objective function score and violation count with the score and violation count of the previous accepted configuration. The best configuration achieved thus far is also recorded. During the melting phase, all new configurations are accepted. During chill periods, a new configuration is accepted if it has less violations than the previously accepted configuration or if it has the same number of violations and a lower score. If the new configuration has a higher score, it is accepted based on the current temperature. The inner loop terminates when the total number of iterations exceeds the number of possible configurations in the search space. During either the melting or chill periods, a new configuration is accepted if its score is lower than the *optimal* score. The optimal score is computed a priori and represents a lower bound for the physical topology solution. If a configuration is found whose score is lower than the optimal, both loops are immediately exited and the SA search algorithm returns with this configuration as the final physical topology solution.

A new configuration is created by randomly selecting an unassigned vnode to be mapped to a pnode. Once all vnodes have pnode assignments, a randomly chosen vnode is selected for reassignment. The chosen vnode has its current pnode assignment removed, and another acceptable pnode or pclass is randomly selected for assignment. If another pnode (pclass) match cannot be found, the chosen vnode is placed back

into the set of unassigned vnodes. Another vnode is then randomly selected, its pnode (pclass) is unassigned, and the second vnode is also placed into the set of unassigned vnodes. The process restarts itself and a randomly chosen unassigned vnode is selected for assignment, since the set of unassigned vnodes is no longer empty. There is a fifty percent chance that the first chosen vnode will be selected again and the probability is greater for a successful match, since there is another pnode resource available.

3.10.2 Modified Anneal.cc using TS Search Algorithm. A single *multimap* data structure implements both the tabu and candidate lists in the TS search algorithm. The multimap data structure is described in the following quote:

> *Multimap* is a Sorted Associative Container that associates objects of type *Key* with objects of type *Data*. Multimap is a Pair Associative Container, meaning that its value type is *pair <const Key, Data>*. It is also a Multiple Associative Container, meaning that there is no limit on the number of elements with the same key. [26]

In the SA search algorithm, a score differential metric establishes the initial temperature that will be used after the melting phase. The score differential is calculated by subtracting the score of the newly accepted configuration from the score of the previously accepted configuration. A score differential of less than zero means the new configuration is of lower quality, increasing the objective function score. TS ignores the SA temperature-based acceptance criteria and accepts all new configurations. The *multimap* data structure in the TS search algorithm uses the score differential as the key for each element. The data portion of the element consists of a second embedded *pair* data structure. The second pair contains the vnode name and the iteration number when the vnode is no longer tabu. The tabu duration is a randomly chosen integer between the values of one and the total number of non-fixed vnodes minus one. The iteration number when the vnode is no longer tabu is the sum of the current iteration number and tabu duration. After all vnodes are initially assigned, a vnode is selected for reassignment by parsing the multimap from beginning to end. The first vnode whose iteration number is less than the current iteration number and is not tabu is chosen for reassignment. After the chosen vnode is successfully mapped, it

is placed back into the multimap with a new tabu iteration number. The insertion operation is never worse than logarithmic [27]. If the chosen vnode cannot be successfully assigned a new pnode, it placed into the set of unassigned vnodes similar to the SA method described above. Another vnode is then randomly selected and its pnode (pclass) is unassigned to increase the chances that the first chosen vnode can be successfully reassigned.

The tabu list is implemented by the iteration number that indicates when the vnode is no longer tabu. The previous configuration will be revisited if the same vnode is immediately reassigned to its prior pnode. The vnode iteration number prevents this from occurring by forcing another vnode to be chosen for reassignment. If all vnodes are tabu, an aspiration condition allows the first vnode in the multimap (the one with the lowest score differential) to be selected for reassignment.

The candidate list is realized since the multimap elements are always sorted in ascending order by score differential key [27]. A low score differential key means the vnode reassignment increased the objective function score and decreased the configuration's quality. Therefore, vnodes with a lower score differential are better candidates to improve the configuration's objective function score than vnodes with a higher score differential.

The TS search algorithm performs a "restart" when the change in the average score drops below a specified epsilon value of 0.000001. This is similar to the SA termination condition, except that SA compares its epsilon value with the change in temperature, not objective function score. TS uses its epsilon value to detect local optimums, when the current objective function score is no better or worse than the configurations immediately before and after. The restart returns TS to the previous best known configuration and the search continues from that point. TS requires that the number of restarts equal the number of non-fixed vnodes prior to ceasing the search. Once the minimum number of restarts has been accomplished, TS uses the epsilon value to detect the local optimum in the current region of space and terminates.

3.11 Summary

An experimental methodology is specified to determine whether a TS implementation of *assign* is superior to Emulab's existing SA implementation in execution time and solution quality when mapping complex virtual topologies. Two primary factors are identified, the type of search algorithm and virtual topology. Both impact *assign's* ability to create a high quality physical topology solution in a minimal amount of time. An experimental design is developed around these primary factors to gauge the performance of both search algorithms, and the results are presented in the next chapter.

IV. Data Analysis

This chapter presents the data collected using the methodology outlined in Chapter 3. The goal of this research is to determine whether a TS implementation of *assign* is superior to Emulab's existing SA implementation with respect to execution time and solution quality. It is expected that a TS implementation of *assign* will locate physical topology solutions in less time than Emulab's existing version of *assign* when mapping identical SF virtual topologies. It is further expected that the number of violations and the objective score of TS solutions will be equal to or lower than SA solutions for the same virtual topologies.

4.1 Validation

Figure 4.1 shows the physical topology produced when the 10-node SF virtual topology is submitted to *assign* using the SA search algorithm and a random number generator seed of 128. *Assign* successfully terminated without violations. The physical solution accurately represents the intended virtual topology, included in the lower right of the diagram for comparison. The 10-node SF virtual topology submitted to the TS version of *assign* produced a similarly accurate physical topology. Many different feasible physical topologies can be produced by either version of *assign* for a given virtual topology, as random number generator seeds are varied. It quickly becomes difficult with large virtual topologies to manually validate each physical topology is an acceptable solution. Given that the mapping process used by *assign* does not change from one virtual topology to the next, the physical topologies produced by both versions of *assign* in this research are assumed valid.

4.2 Vlink Multiplexing Issues

The physical solution shown in Figure 4.1 demonstrates some key resource conversion techniques. Four vnodes are multiplexed onto two pnodes. The physical topology is not spread across multiple switches, unnecessarily using limited inter-switch bandwidth. The solution is not optimal, however, since a better solution would

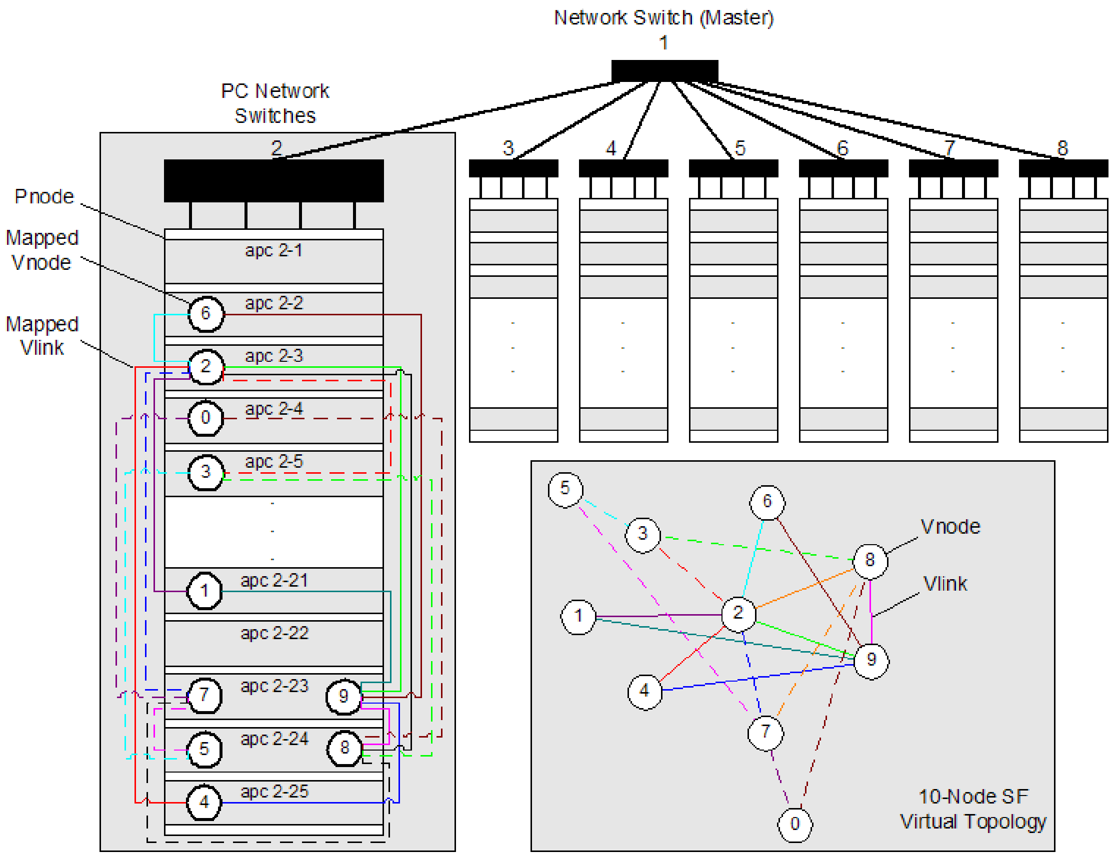
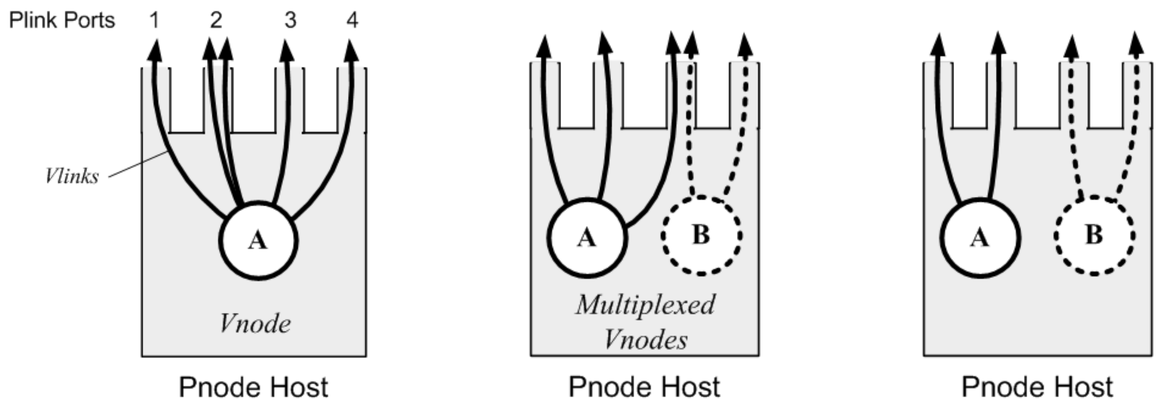


Figure 4.1: A physical topology solution for the 10-node SF virtual topology submitted to the SA version of *assign*. The random number generator seed is 128. The entire physical topology resides on pnodes connected to PC network switch 2. Only pnodes that contain vnodes are shown. Vnodes are mapped one-to-one to pnodes except for vnodes 7 and 9 mapped to pnode apc2-23 and vnodes 5 and 8 mapped to pnode apc2-24. All vlinks are mapped one-to-one onto intra-switch plinks. The mapped vlinks are color-coded and not shown passing through network switch 2 for easy comparison to the 10-node SF virtual topology on the right.

incorporate intra-node plinks and multiplex more than one vlink onto an intra-switch plink. Instead, all vlinks are mapped one-to-one using separate intra-switch plinks. The mapped vlinks are color-coded in Figure 4.1 and not shown passing through network switch 2 so they can be more easily compared to the 10-node SF virtual topology on the right.

While vlink multiplexing across inter-switch plinks was a common occurrence, the lack of intra-switch vlink multiplexing and intra-node plinks was apparent throughout testing. No instances of intra-node plinks or intra-switch vlink multiplexing could



(a) (b) (c)

Figure 4.2: Three examples illustrating vlink multiplexing in a pnode host. The large box represents the pnode host and the channels at the top represent available plinks. Vnodes are depicted by circles and vlinks are shown by lines with arrows originating from the circles. *Assign's* mapping precheck prevents condition (a) from occurring, even though plink port 2 has enough bandwidth to support both vlinks. Condition (b) was not observed in any of the experiments completed for this research, even for plinks that had enough bandwidth to support vlink from different vnodes. The only acceptable mapping is condition (c).

be found in any of the physical topologies generated by both search algorithms. Figure 4.2 illustrates three cases of vlink multiplexing in a pnode host. Condition (a) in Figure 4.2 is prevented by *assign's* mapping prechecks. The original 200 to 1000-node SF virtual topologies did not pass *assign's* mapping prechecks because the nine topologies each had a small number of vnodes with greater than 20 vlinks, even though each plink is capable of supporting multiple vlinks. As an example, the vnode mapping precheck indicated that no possible pnode assignments existed for three of the vnodes in the 200-node SF topology. Condition (b) was never observed in any of the physical topologies produced by the search algorithms while condition (c) was commonplace. Figure 4.1 demonstrates two cases of condition (c) where vnodes 7 and 9 are mapped together on a single pnode and vnodes 5 and 8 are mapped together on a different pnode.

Vnodes with 20 or more vlinks in the SF virtual topologies with 200-nodes or greater were modified to pass *assign's* mapping prechecks. Table 4.1 shows the

Table 4.1: Vnodes from the nine SF virtual topologies that were reduced to 20 vlinks in order to pass *assign's* mapping prechecks. The first column identifies the affected SF virtual topology. The second column lists the number of vnodes in the topology with more than 20 links. The third column specifies the original number of vlinks for each vnode in the second column.

SF Virtual Topology	**Number of Vnodes With Over 20 Vlinks**	**Original Number of Vlinks**
200	3	26, 22, 21
300	4	22, 35, 25, 30
400	7	23, 48, 21, 28, 24, 22, 31
500	7	26, 44, 21, 28, 29, 54, 47
600	9	39, 39, 28, 43, 47, 40, 33, 24, 26
700	12	22, 29, 23, 54, 34, 40, 36, 26, 23, 26, 23, 29
800	13	21, 26, 46, 22, 27, 47, 31, 31, 55, 27, 57, 24, 33
900	13	59, 22, 31, 38, 29, 42, 50, 37, 32, 44, 31, 48, 22
1000	12	29, 22, 21, 31, 21, 29, 111, 30, 21, 42, 30, 84

affected vnodes from the SF virtual topologies and their original vlink count. All the vnodes in Table 4.1 had their vlink total reduced to 20 by eliminating vlinks from the .top files. For example, in the 500-node virtual topology, the first six vlinks where the vnode with 26 vlinks was the source or destination were deleted. Care was taken to ensure no vnodes were isolated after all necessary vlinks were removed. This removal process constituted an 11 percent average reduction in the total amount of vlinks in affected SF virtual topologies.

The restrictions on vlink multiplexing created another problem. If vlinks can only be mapped one-to-one onto intra-node plinks, then a feasible physical topology no longer exists for SF virtual topologies with 200 nodes or greater using the original set of testbed resources. The original .ptop file incorporated only five pnodes with 20 plinks (see Table 3.1). As shown in Table 4.1, all SF virtual topologies with 400

nodes or greater have at least seven vnodes with 20 vlinks. Additionally, the 200 and 300-node SF virtual topologies each have multiple vnodes with over 15 vlinks preventing a solution using the original set of testbed resources. Another .ptop file was fashioned representing a second set of testbed resources. All 525 pnodes in this set of testbed resources have 20 gigabit plinks connecting to their corresponding PC network switch.

4.3 Analysis of Valid Solutions

The primary performance metric for this study is whether a feasible physical topology solution is produced from the provided virtual topology and the set of available testbed resources. SA and TS are sub-optimal search algorithms, hence a feasible physical topology solution is not guaranteed every time the search algorithms have completed. The goal of this research is to determine whether a TS implementation of *assign* is superior to Emulab's existing SA implementation in terms of execution time and solution quality. In order to meet this goal, physical topologies produced by SA and TS that are an accurate representation of the submitted virtual topologies (e.g., feasible solutions) must be available for comparison. Tables 4.2 and 4.3 list the number of valid physical topology solutions created by both search algorithms for all virtual topologies. 200 trials were run for each virtual topology and search algorithm combination. The first ten rows of Tables 4.2 and 4.3 show the results when the 10 to 100-node virtual topologies are mapped to the original set of testbed resources. The last nine rows of both tables show the results when the 200 to 1000-node topologies (altered to pass *assign* prechecks) are mapped to the second set of testbed resources described in the previous section.

Tables 4.2 and 4.3 also show the results of a two binomial proportion test on the number of valid physical topology solutions produced by both search algorithms. A two binomial proportion test with 95 percent confidence was conducted to determine if there is a statistically significant improvement in the number of valid solutions when using TS versus SA. The null hypothesis for the proportion test is that the proportion

Table 4.2: The number of valid physical topology solutions created by the search algorithms for the random virtual topologies. 200 trials were run for each virtual topology and search algorithm combination. The second set of testbed resources was used for virtual topologies with 200 nodes and greater (indicated by a single asterisk). A double asterisk indicates Fisher's exact test was used to calculate the 2-Proportion test p-value, as the sample size was to small for normal approximation.

Random Virtual Topology	**SA Valid Solutions**	**TS Valid Solutions**	**2-Proportion Test p-value**	**Higher Proportion**
10	200	200	n/a	No difference
20	200	200	n/a	No difference
30	200	200	n/a	No difference
40	200	200	n/a	No difference
50	200	200	n/a	No difference
60	196	198	0.685**	No difference
70	170	190	0.001	TS
80	200	176	0.0	SA
90	14	164	0.0	TS
100	0	88	0.0	TS
200*	166	200	0.0	TS
300*	30	200	0.0	TS
400*	30	200	0.0	TS
500*	0	200	0.0	TS
600*	14	200	0.0	TS
700*	12	200	0.0	TS
800*	0	200	0.0	TS
900*	5	200	0.0	TS
1000*	0	200	0.0	TS

of valid solutions produced by TS and SA for a given network type and number of vnodes do not differ by a statistically significant amount. The alternate hypothesis is that the TS proportion is greater than the SA proportion of valid solutions (upper-tailed). If the p-value shown in the fourth column of Tables 4.2 and 4.3 is smaller than the α value of 0.05 (95 percent confidence), then the result is consistent with alternate hypothesis. For random virtual topologies shown in Table 4.2, there was no difference between the search algorithms for topologies with 60 nodes or less. For random topologies with greater than 60 nodes, there was a statistically significant

Table 4.3: The number of valid physical topology solutions created by the search algorithms for the SF virtual topologies. 200 trials were run for each virtual topology and search algorithm combination. The second set of testbed resources was used for virtual topologies with 200 nodes and greater (indicated by a single asterisk).

SF Virtual Topology	SA Valid Solutions	TS Valid Solutions	2-Proportion Test p-value	Higher Proportion
10	200	200	n/a	No difference
20	200	200	n/a	No difference
30	200	200	n/a	No difference
40	200	200	n/a	No difference
50	130	200	0.0	TS
60	18	194	0.0	TS
70	0	138	0.0	TS
80	0	48	0.0	TS
90	0	148	0.0	TS
100	0	106	0.0	TS
200*	0	0	n/a	No difference
300*	0	82	0.0	TS
400*	0	0	n/a	No difference
500*	0	0	n/a	No difference
600*	0	96	0.0	TS
700*	0	0	n/a	No difference
800*	0	140	0.0	TS
900*	0	22	0.0	TS
1000*	0	46	0.0	TS

improvement in the number of valid physical topologies produced when using TS. The only exception was the 80-node random topology, when there was a statistically significant improvement using SA. For the SF virtual topologies shown in Table 4.3, there was no difference between the search algorithms for topologies with 40 nodes or less. For SF topologies with greater than 40 nodes, there was a statistically significant improvement in the number of valid physical topologies produced when using TS for all but four topologies. There was no difference between the search algorithms for the SF topologies of 200, 400, 500 and 700 nodes, as neither search algorithm could produce a valid solution. Figures 4.3 and 4.4 visually display the data from Tables 4.2 and 4.3 as line graphs for easy reference.

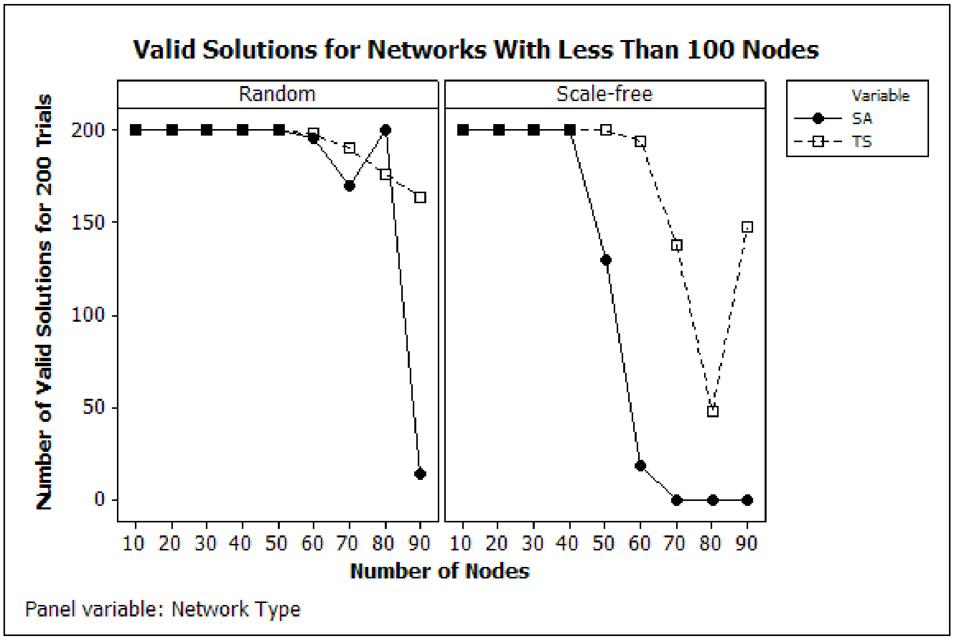


Figure 4.3: Line graph of the number of valid solutions produced by SA and TS for random and SF networks with less than 100 nodes. TS was able to produce a statistically significant greater amount of solutions for all 18 virtual topologies with the exception of the 80-node random topology. SA was able to produce a statistically significant greater amount of solutions for this topology.

Figure 4.3 shows that TS and SA are able to produce a valid solution for every trial for random virtual topologies with 50 nodes or less and SF virtual topologies with 40 nodes or less. In both networks, SA starts to fail to produce valid solutions for networks with fewer vnodes than TS. Additionally, the rate of failure is greater for SA than TS. The exception is the 80-node random topology, where SA is able to produce a higher amount of valid solutions. In Figure 4.4, SA and TS are able to produce a greater proportion of valid solutions for the 200 than the 100-node random topology. This is because the second set of testbed resources is used for topologies greater than 100 nodes. Figure 4.4 also shows that TS is able to find a valid solution for every trial for random topologies with 200 nodes and greater, while the proportion of valid solutions produced by SA drops off quickly for random topologies with greater than 200 nodes. SA is never able to find a valid solution for SF topologies with 100 nodes or greater.

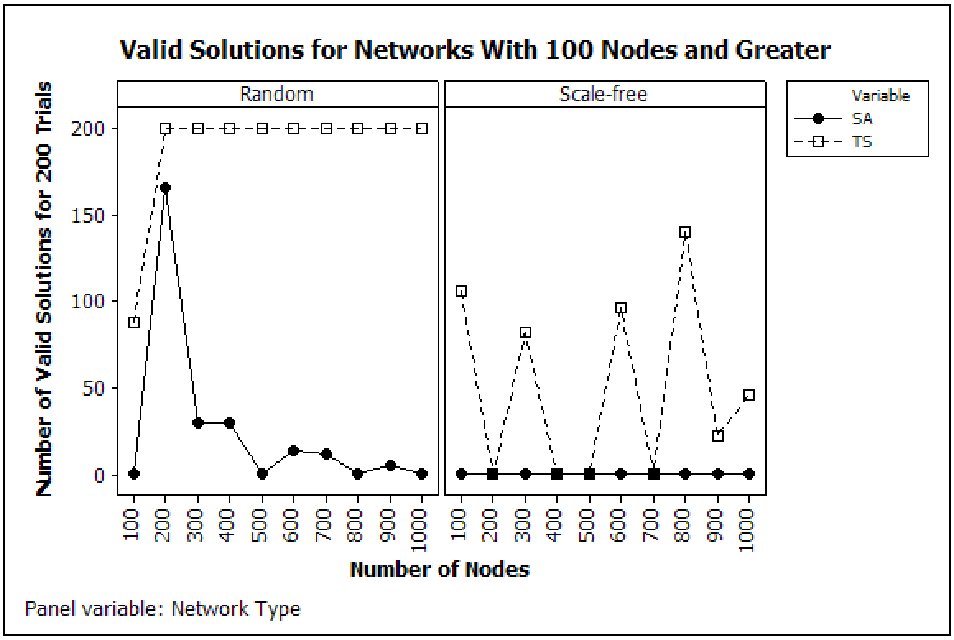


Figure 4.4: Line graph of the number of valid solutions produced by SA and TS for random and SF networks with 100 nodes or greater. TS was able to produce a statistically significant greater amount of solutions for all 20 virtual topologies with the exception of four topologies. There was no statistically significant improvement between the search algorithms for the SF topologies of 200, 400, 500 and 700 nodes. Neither search algorithm could produce a valid solution for these topologies.

TS is able to find valid solutions for a portion of the SF virtual topologies with 100 nodes and greater. However, this trend is not linear and seems to oscillate in Figure 4.4. This anomaly is also present in the 80-node SF virtual topology shown in Figure 4.3, which is mapped with a lower proportion of success by TS than the 90 and 100-node SF virtual topologies. Due to a uniform degree distribution among nodes, the structure of random virtual topologies do not vary to the extent of the SF topologies. Figures 4.3 and 4.4 show that for the majority of the random virtual topologies, as the number of nodes increases, the proportions of valid solutions created by both search algorithms decreases. Figure 4.4 shows that node number is not always the dominate factor in predicting whether TS is able to produce a valid solution in the case of SF virtual topologies. Other SF characteristics, such as the number of hubs and the average number of links per hub, may also influence the ability of TS to produce valid solutions.

4.4 Analysis of Execution Time and Solution Quality

Figures 4.5 and 4.6 show the measured execution time and objective function score for all virtual topologies and search algorithms. Only the first 20 of the 200 trials is required for analysis, due to the small variance in data values for the same virtual topology and search algorithm combination. The measured objective function scores, violation count and execution times for the first 20 trials for all virtual topology and search algorithm combinations is recorded in Appendix C. Only 10 of the 19 virtual topologies in which both search algorithms were able to produce at least one valid solution are analyzed. Trials that did not produce a valid solution were excluded. A total of 554 data points (approximately 27 data points per search algorithm and virtual topology combination) were used in the analysis. Figure 4.5 shows that SA maintains a relatively constant execution time of 1 to 2 seconds regardless of network type and node number. Conversely, TS execution time increases with node number, especially for random virtual topologies greater than 100 nodes. Figure 4.6 shows that both SA and TS objective function score increases with node number. However, the rate at which objective score increases is steeper for TS than SA.

Figures 4.7 through 4.10 show the residuals for measured execution times and objective function scores for both search algorithms. In all four figures, the "Normal Probability Plot of the Residuals" and the "Histogram of the Residuals" show that errors are not normally distributed. The "Residuals Versus the Fitted Values" show errors do not have a constant standard deviation, as there appears to be a tendency of either increasing or decreasing spread as response increases towards the left of the graph. Only in Figure 4.7 do errors appear to be independent, as the "Residuals Versus the Order of the Data" plot in the remaining three figures show trends in the graphs. An Analysis of Variance (ANOVA) cannot be conducted to determine which search algorithm performs better in terms of execution time or objective function score, as the underlying ANOVA assumptions of statistically independent errors, normally distributed errors and errors with a constant standard deviation are violated.

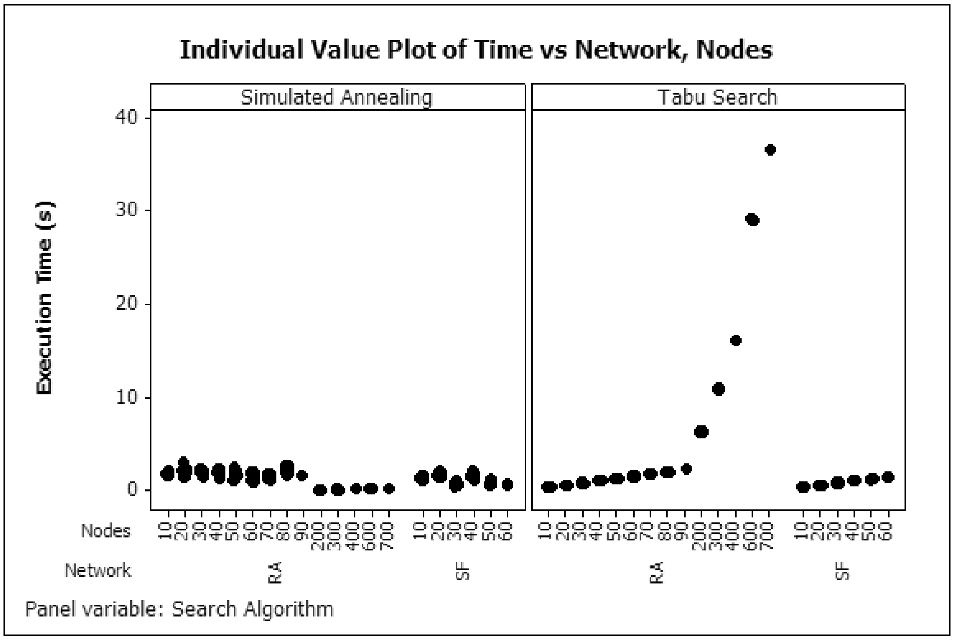


Figure 4.5: Measured execution time by search algorithm and virtual topology. SA maintains a relatively constant execution time of 1 to 2 seconds regardless of network type and node number. TS execution time increases with node number, especially for random virtual topologies greater than 100 nodes.

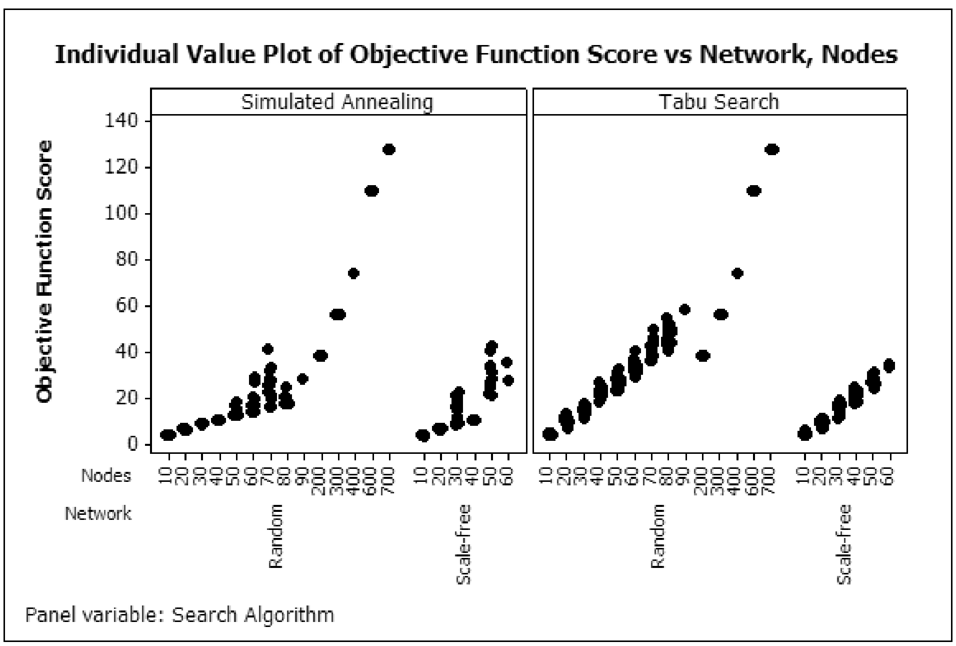


Figure 4.6: Measured objective function score by search algorithm and virtual topology. Both SA and TS objective function score increase with node number, however, the rate of increase is greater for TS than SA.

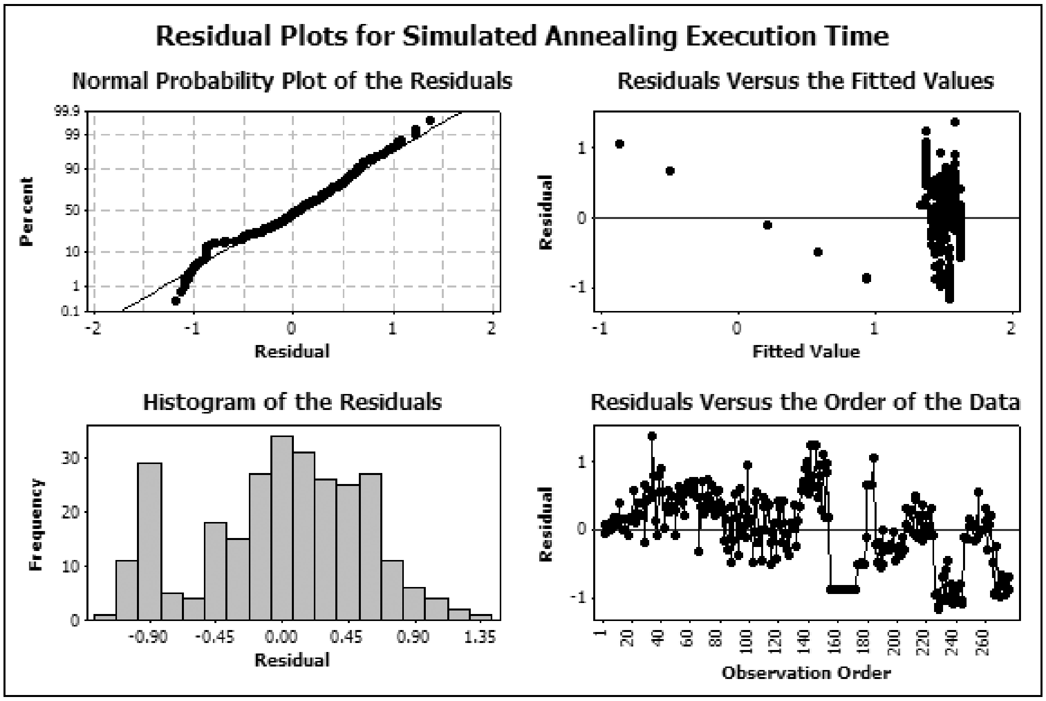


Figure 4.7: Residual plots for measured simulated annealing execution time data.

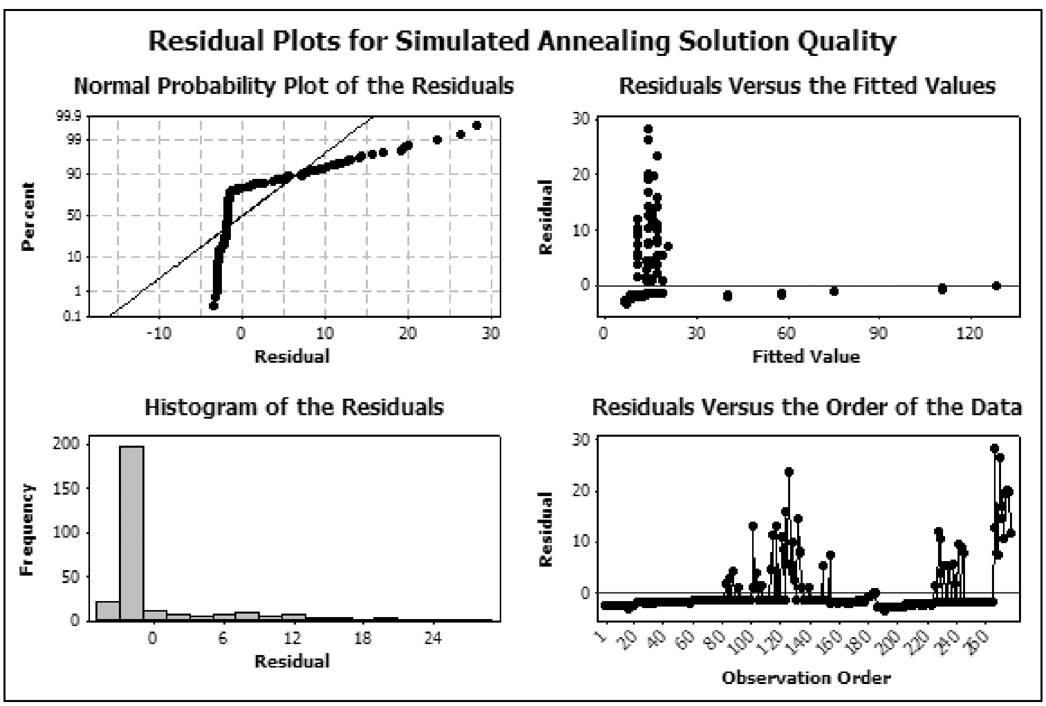


Figure 4.8: Residual plots for measured simulated annealing objective function score data.

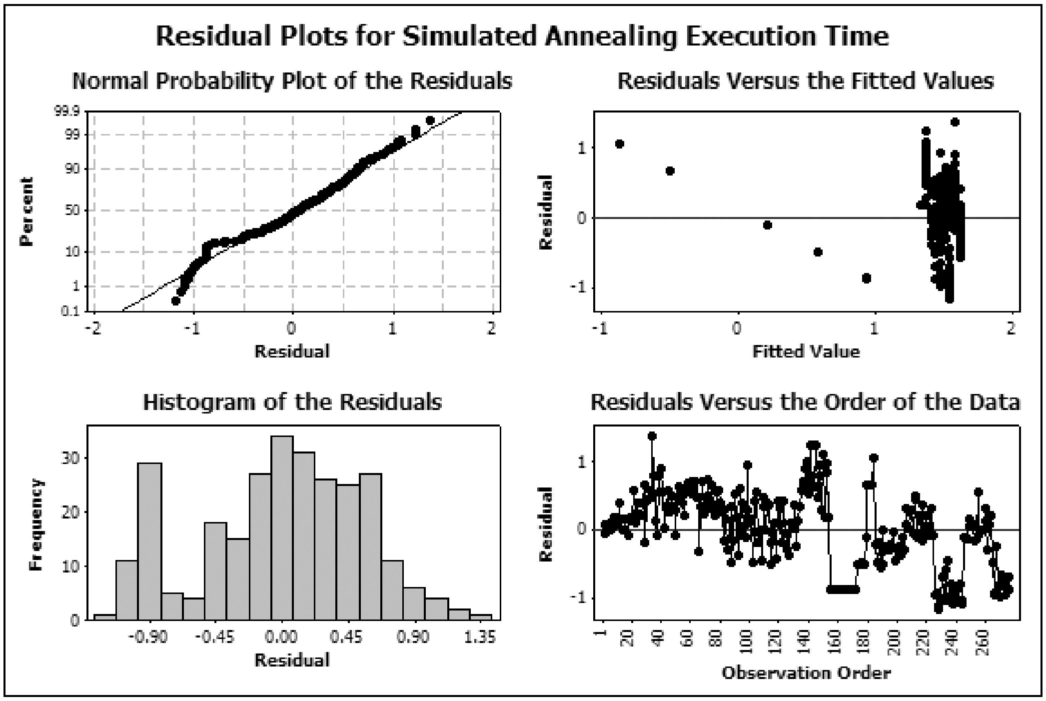


Figure 4.9: Residual plots for measured tabu search execution time data.

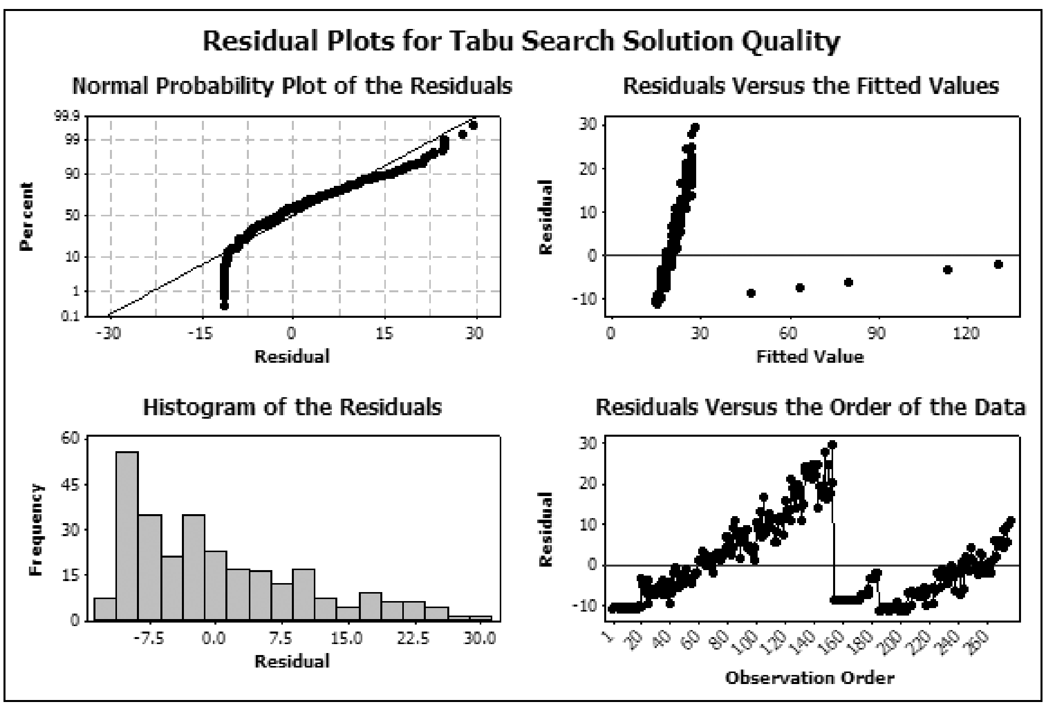


Figure 4.10: Residual plots for measured tabu search objective function score data.

4.4.1 Analysis of Execution Time. Figures 4.11 through 4.14 show the 95 percent confidence intervals for the mean execution time for both search algorithms with all virtual topologies combined, separated by network type and separated by the number of nodes. When all virtual topologies are combined in Figure 4.11, the execution time of TS is statistically higher than SA. However, when the virtual topologies are separated by network type in Figure 4.12, the execution time of TS is statistically lower than SA for SF virtual topologies. The execution time of TS is statistically higher than SA for virtual topologies with 100 nodes and greater, as shown by Figure 4.14. For virtual topologies with less than 100 nodes, the execution time of TS is statistically lower than SA, as shown in Figure 4.14. These observations conflict with the first research hypothesis, that a TS implementation of *assign* could locate physical topology solutions in less time than Emulab's existing version of *assign* when mapping identical virtual topologies. Only in cases of SF virtual topologies and topologies with 100 nodes and greater was TS able to execute in less time than SA.

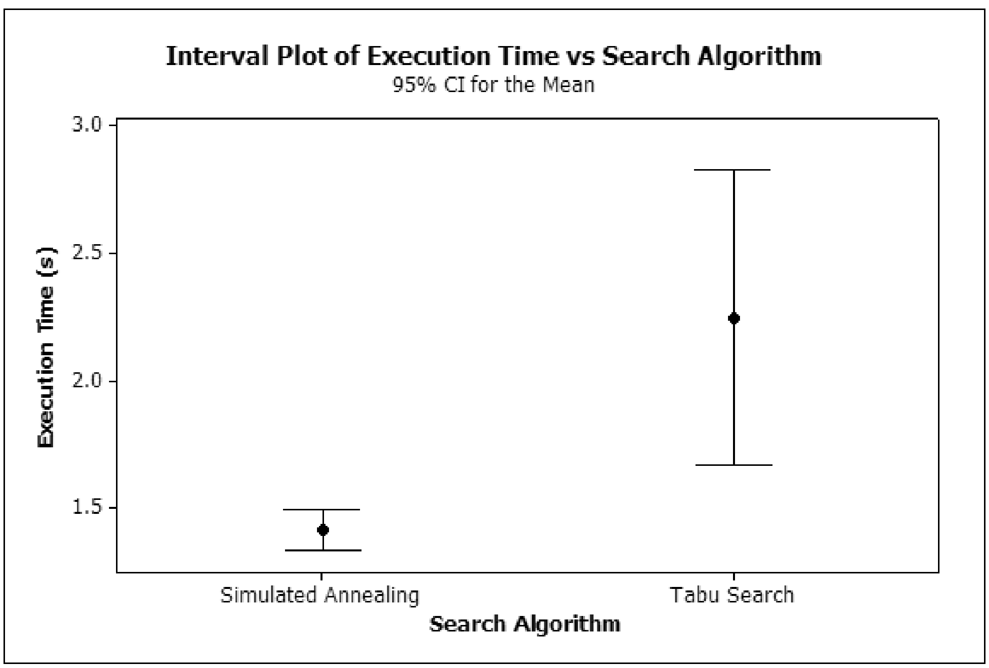


Figure 4.11: 95 percent confidence intervals for the mean execution time for both search algorithms with all virtual topologies combined, regardless of network type and node number.

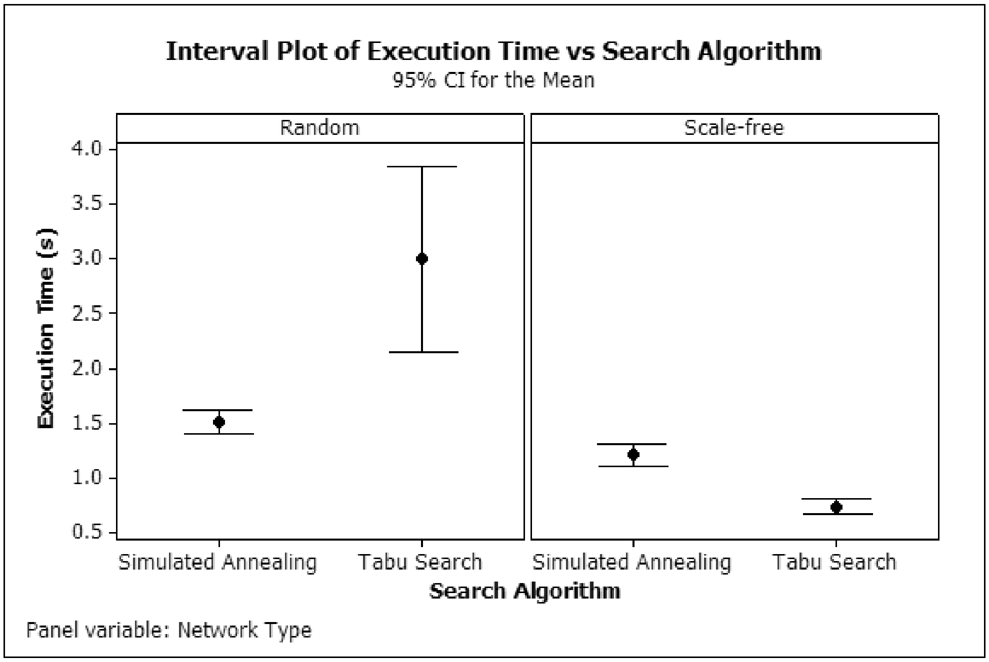


Figure 4.12: 95 percent confidence intervals for the mean execution time for both search algorithms for random and SF virtual topologies.

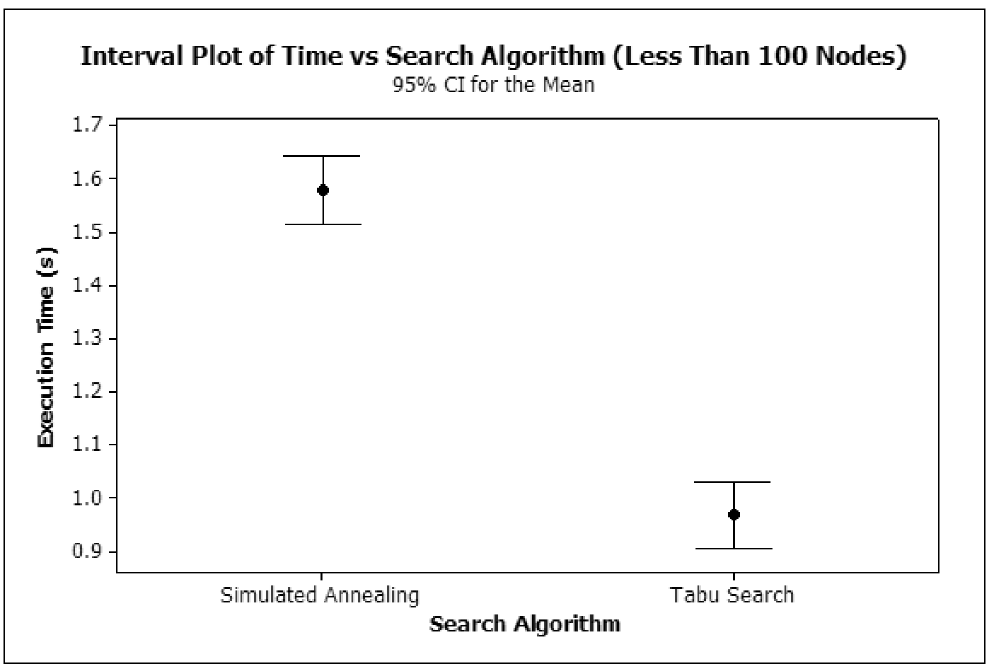


Figure 4.13: 95 percent confidence intervals for the mean execution time for both search algorithms for virtual topologies with less than 100 nodes.

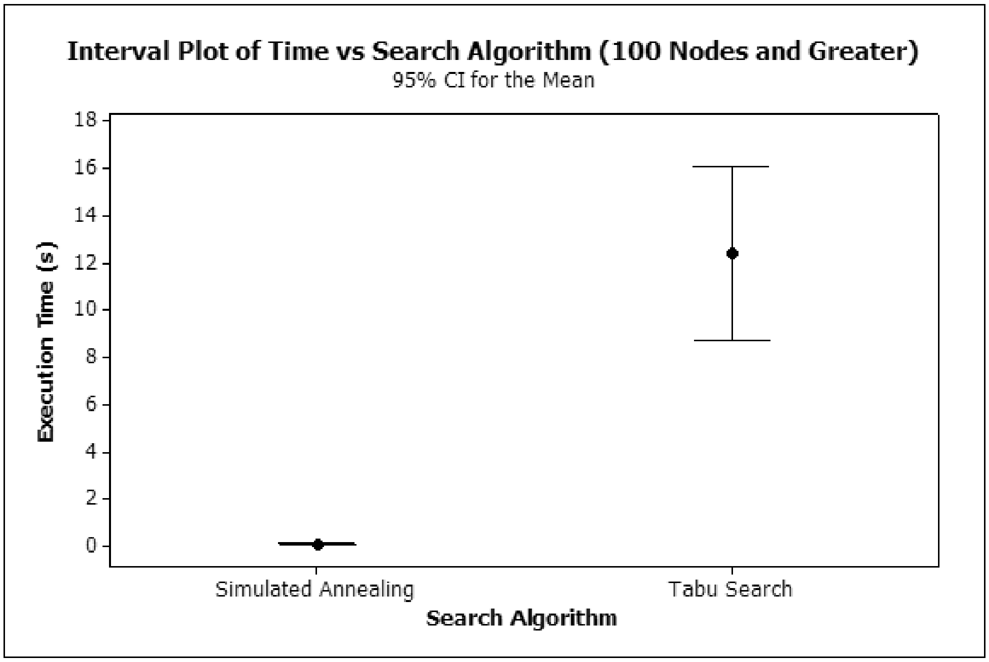


Figure 4.14: 95 percent confidence intervals for the mean execution time for both search algorithms for virtual topologies with 100 nodes and greater.

4.4.2 Analysis of Objective Function Score. Figures 4.15 through 4.18 show the 95 percent confidence intervals for the mean objective function score for both search algorithms with all virtual topologies combined, separated by network type and separated by the number of nodes. When all virtual topologies are combined in Figure 4.15, the objective function score of TS is statistically higher than SA. When virtual topologies are categorized by network type in Figure 4.16, there is no statistically significant difference in score for TS and SA for SF topologies. In cases when the number of nodes in the virtual topology is less than 100, there is no statistically significant difference in the score of both search algorithms. SA does produce a statistically lower score for virtual topologies with 100 nodes or greater. These observations conflict with the second research hypothesis, that the number of violations and the objective score of TS solutions would be equal to or lower than SA solutions for the same virtual topologies. Only for SF virtual topologies and topologies with less than 100 nodes is there no statistically significant difference in the objective function score of both search algorithms.

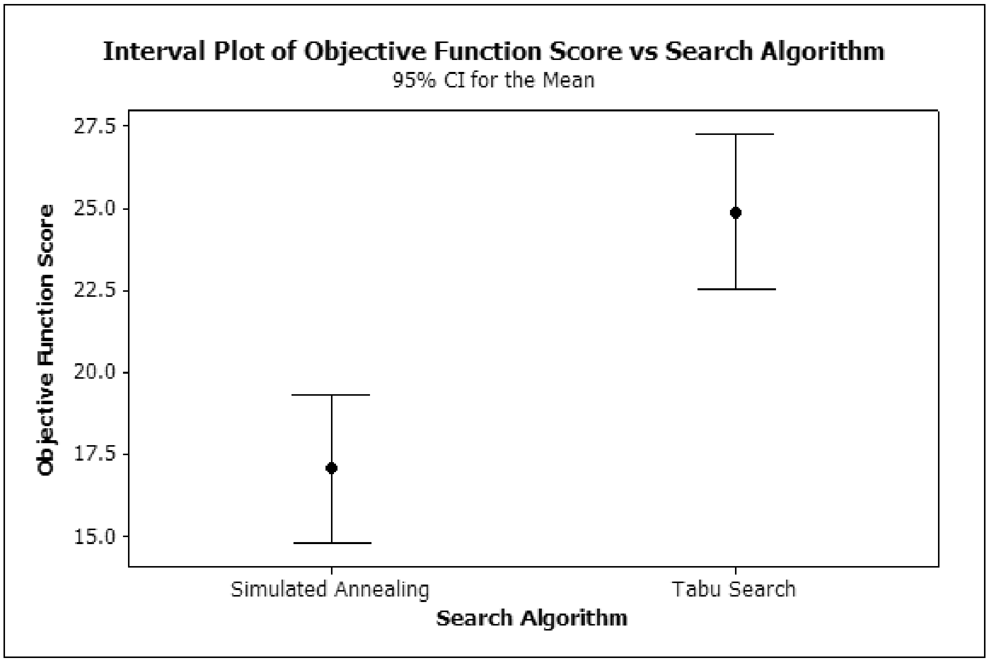


Figure 4.15: 95 percent confidence intervals for the mean objective function score for both search algorithms with all virtual topologies combined, regardless of network type and node number.

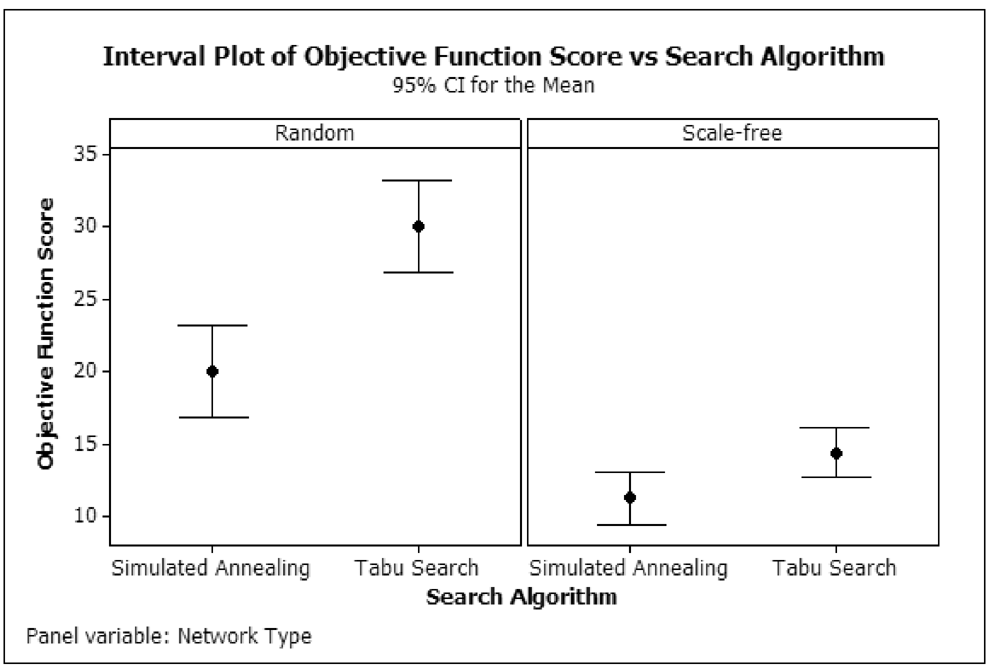


Figure 4.16: 95 percent confidence intervals for the mean objective function score for both search algorithms for random and SF virtual topologies.

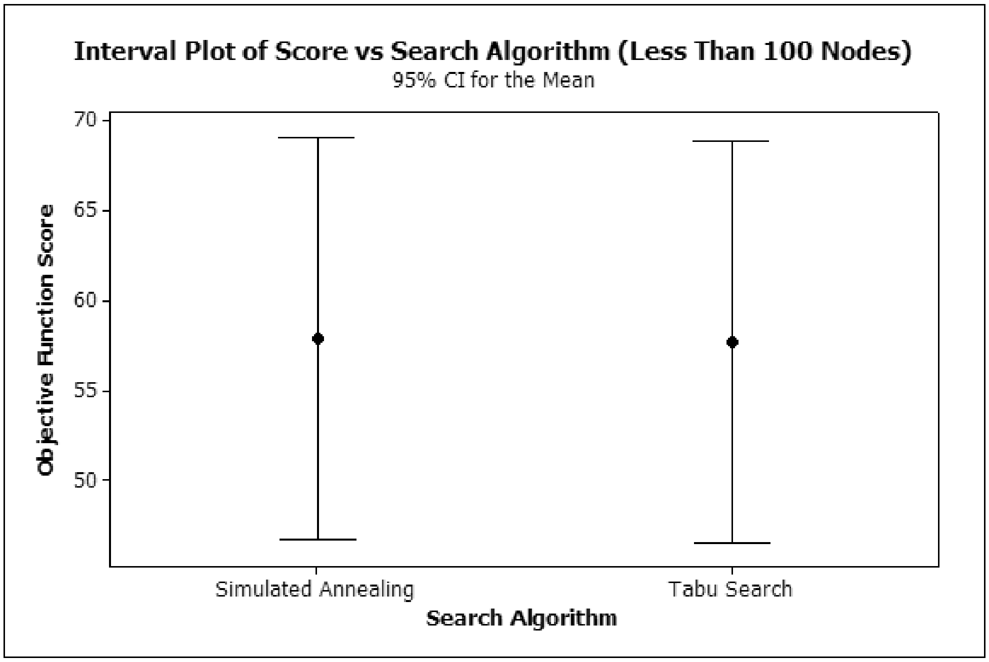


Figure 4.17: 95 percent confidence intervals for the mean objective function score for both search algorithms for virtual topologies with less than 100 nodes.

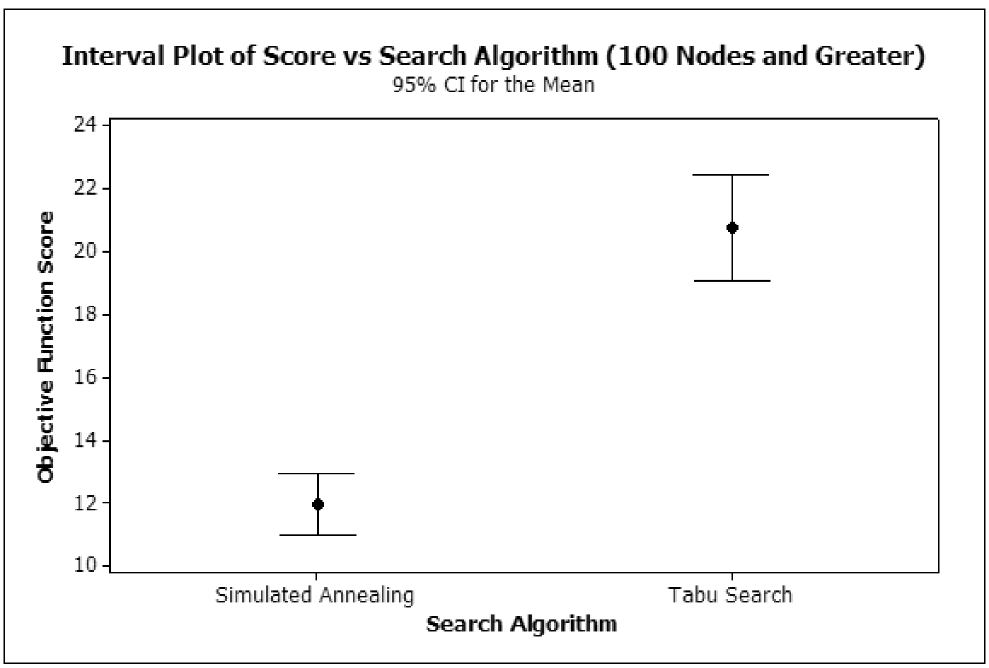


Figure 4.18: 95 percent confidence intervals for the mean objective function score for both search algorithms for virtual topologies with 100 nodes and greater.

4.5 Summary

This chapter presented the measured data for the execution time and objective function scores for SA and TS search algorithms. The first analysis conducted a two binomial proportion test to determine which search algorithm was able to produce a higher proportion of valid physical topology solutions. The second analysis determined which search algorithm was able to execute in less time for all combined virtual topologies, and then for virtual topologies divided by network type and node number. The third analysis determined which search algorithm was able to produce a lower objective function score for similar virtual topology categories. In the next chapter, conclusions drawn from these analyses are discussed along with suggestion for future research.

V. Conclusions

This chapter presents a summary of the research conducted and conclusions from the analysis provided in Chapter 4. Significance of the research, contributions and areas for future research is discussed.

5.1 Research Summary and Conclusions

This research investigated the problem of creating high quality, feasible solutions for complex networks with thousands of nodes in a minimum amount of time. The goal of this research was to determine whether a TS implementation of *assign* was superior to Emulab's existing SA implementation in terms of execution time and solution quality. The first hypothesis was that a TS implementation of *assign* could locate physical topology solutions in less time than Emulab's existing version of *assign* when mapping identical virtual topologies. The second hypothesis was that the number of violations and the objective score of TS solutions would be equal to or lower than SA solutions for the same virtual topologies.

The first hypothesis was proven to be false because only in cases of SF virtual topologies and topologies with 100 nodes and greater was TS able to execute in less time than SA. The second hypothesis was also proven to be false because only for SF virtual topologies and topologies with less than 100 nodes was there no statistically significant difference in the objective function score of both search algorithms. It should be noted that virtual topologies of 100 nodes and greater included only random networks, as SA was unable to produce a valid solution for SF networks with over 70 nodes. The fact that TS was able to execute quicker and produce equivalent objective function scores for SF virtual topologies should also be noted. It stands to reason that the random or lack of structure in the random virtual topologies was better match for simulated annealing, while TS was the better search algorithm for the more defined structure of the SF topologies.

Although TS was unable to best SA in terms of execution time and solution quality, it was able to produce an equal higher proportion of valid physical solutions

for all 38 virtual topologies except for one. This was especially prominent for virtual topologies with 100 nodes or greater. As the number of nodes is dominant feature of complex networks, the success of TS over SA in simply producing a valid physical topology solution should not be overlooked.

5.2 Research Significance and Contributions

This is the first known implementation of TS in a solver for the testbed mapping problem. Previous research focused on the use of external programs, such as METIS, to augment *assign's* existing SA search algorithm in the creation of valid physical topologies. Although METIS was able to execute very quickly, it caused a higher use of resources per virtual topology as a tradeoff for its speed. Additionally, the SA search algorithm still needed to execute in order to create a physical topology solution. Genetic algorithms were also explored as SA alternatives in journals and other published research papers. None of the genetic algorithms developed were able to best SA in terms of execution time or solution quality. The significance of this research is that a fully realizable version of *assign* has been produced that uses TS. This version of *assign* can be compiled and immediately put into use on an Emulab testbed.

5.3 Future Work

Assign has many areas available for future research. An analysis of the impacts of different termination conditions would benefit both the SA and TS search algorithms, as shown by the additional data provided in the tables of Appendix C. It appears that in many cases SA terminated its search prematurely when attempting to map the larger virtual topologies. Also, since the tabu list duration impacts the performance of the TS search algorithm to such a great degree, development of a reactive TS search algorithm that automatically adapts its tabu duration based on search performance may yield a better performing search algorithm than this implementation of TS.

Appendix A. Makefile Used to Compile Assign Source Code

Listing A.1 details the makefile configuration used to analyze the performance of *assign* using SA and TS search algorithms. The makefile and parameters indicated in Table 3.3 comprise the software test environment. The makefile includes the compiler optimization level, cooling schedule and other SA-specific options, and settings to change vlink mapping behavior.

Listing A.1: *Assign* Makefile

```
#
# EMULAB-COPYRIGHT
# Copyright (c) 2000-2005 University of Utah and the Flux Group.
# All rights reserved.
#
SRCDIR          = .
TESTBED_SRCDIR  = ..
OBJDIR          = ..
SUBDIR          = assign
MAKEFILE_IN     = ./GNUmakefile.in

include $(OBJDIR)/Makeconf

# Uncomment to build with GCC version 3.3.x
#CC=gcc33
#CPP=cpp33
#CXX=g++33

all: assign

include $(TESTBED_SRCDIR)/GNUmakerules

OBJS=parse_top.o parse_ptop.o assign.o pclass.o vclass.o \
     config.o score.o parser.o solution.o anneal.o \
     featuredesire.o neighborhood.o fstring.o
LIBS+= -lm
LDFLAGS+= -pipe -O3
CXXFLAGS = -pipe -I/usr/local/include -ftemplate-depth-40

# Sets compiler optimization to level 3
CXXFLAGS += -O3
# Sets compiler to optimization to level 0, includes
# verbose warnings and implements debugger
#CXXFLAGS += -O0 -g -Wall -DVERBOSE
# Various assign diagnostic options
#CXXFLAGS += -DSCORE_DEBUG
#CXXFLAGS += -DSCORE_DEBUG_MORE
#CXXFLAGS += -DPCLASS_DEBUG
#CXXFLAGS += -DDUMP_GRAPH
#CXXFLAGS += -DSCORE_DEBUG_LOTS
```

```
#CXXFLAGS += -DSTATS

# Assign now supports a dizzing array of defines.
# Here are the ones used for a typical build:

# Pick cooling schedule
CXXFLAGS += -DMELT -DEPSILON_TERMINATE -DCHILL -DNEIGHBOR_LENGTH \
            -DLOCAL_DERIVATIVE -DALLOW_NEGATIVE_DELTA
# Bug/scoring fixes
CXXFLAGS += -DINTERSWITCH_LENGTH -DPNODE_SWITCH_LOAD \
              -DFIX_SHARED_INTERFACES
# Various tweaks to the simulated annealing behavior
CXXFLAGS += -DFIND_PNODE_SEARCH -DNO_REVERT
# Keeps information about which pclasses are potential mappings
# for vnodes on a per-vnode basis, not a per-type basis
CXXFLAGS += -DPER_VNODE_TT
# Should be on by default, but not well tested enough
#CXXFLAGS += -DSMART_UNMAP
# Make sure that all emulated links that are assigned to a plink
# have the same endpoints
CXXFLAGS += -DFIX_PLINK_ENDPOINTS
# Allow pnodes to cap the amount of trivial link bandwidth
# they can handle
CXXFLAGS += -DTRIVIAL_LINK_BW
# Use the old acceptance criteria, which gives special treatment
# to violations
CXXFLAGS += -DSPECIAL_VIOLATION_TREATMENT

# If you're looking to turn on or off USE_OPTIMAL, its now a
# cmdline option. Use OP={0,1} on the command line at run time.

DEPLIBS=$(OBJS)

assign: ${MAKEFILE_IN} ${DEPLIBS} ${OBJS}
        ${CXX} -o assign ${LIBS} $(OBJS) ${LDFLAGS}

install: $(INSTALL_LIBEXECDIR)/assign

clean:
        -${RM} *.o assign

# All of this generated with 'g++ -MM' - to make automatic,
# since none of it ever changes
anneal.o: anneal.cc anneal.h port.h delay.h physical.h common.h \
 config.h featuredesire.h pclass.h virtual.h maps.h score.h \
 solution.h vclass.h ${MAKEFILE_IN}
assign.o: assign.cc port.h common.h config.h delay.h physical.h \
 featuredesire.h virtual.h vclass.h pclass.h score.h solution.h \
 maps.h anneal.h ${MAKEFILE_IN}
config.o: config.cc config.h ${MAKEFILE_IN}
featuredesire.o: featuredesire.cc featuredesire.h common.h \
 config.h ${MAKEFILE_IN}
```

```
parse_ptop.o: parse_ptop.cc port.h delay.h physical.h common.h \
 config.h featuredesire.h parser.h ${MAKEFILE_IN}
parse_top.o: parse_top.cc port.h common.h config.h vclass.h \
 delay.h physical.h featuredesire.h virtual.h parser.h anneal.h \
 pclass.h ${MAKEFILE_IN}
parser.o: parser.cc parser.h port.h ${MAKEFILE_IN}
pclass.o: pclass.cc port.h common.h config.h delay.h physical.h \
 featuredesire.h virtual.h pclass.h ${MAKEFILE_IN}
score.o: score.cc port.h common.h config.h vclass.h delay.h \
 physical.h featuredesire.h virtual.h pclass.h score.h \
 /usr/include/math.h ${MAKEFILE_IN}
solution.o: solution.cc solution.h port.h delay.h physical.h \
 common.h config.h featuredesire.h virtual.h maps.h vclass.h \
 ${MAKEFILE_IN}
vclass.o: vclass.cc port.h common.h config.h vclass.h delay.h \
 physical.h featuredesire.h virtual.h ${MAKEFILE_IN}
```

Appendix B. Virtual Topology and Testbed Resource Input Files

This appendix describes the format of the .ptop and .top text files that are inputs for *assign*. The .ptop text file defines the set of available testbed resources and is described in the first section. The second section outlines the structure of .top file that specifies the SF virtual topologies. The .ptop and .top file format descriptions are from [31], a readme file included with the *assign* source code. Otter visualizations for the 10, 50, 100 and 1000-node SF virtual topologies are also provided.

B.1 Set of Available Testbed Resources (.ptop file)

Each line in the .ptop file describes either a pnode or a plink. Lines describing pnodes are in the format:

```
node <node> <types> [- <features>]
```

<node> is the string identifier of the pnode.

<types> is a space-separated list of <type>:<number>.

<type> is the string identifier for the vnode types this pnode can host. “switch” is a special identifier that indicates the vnode type is a network switch.

<number> is the number of vnodes of the particular type this pnode can host.

<features> is a space-separated list of <feature>:<cost>.

<feature> is the string identifier of the feature.

<cost> is the cost of the feature being wasted.

Lines designating plinks are in the following format:

```
link <link> <src>[:<smac>] <dst>[:<dmac>] <bw> <delay> <loss> <num>
<type>
```

<link> is the string identifier for the plink.

<src>,<dst> are source and destination pnodes.

<smac>,<dmac> are optional arguments that are medium access control addresses or other strings to distinguish pnode ports. If omitted, the string “(null)” is used. These arguments are not present on interswitch plinks.

<bw>,<delay>,<loss> are the characteristics of the plink.

<num> is the number of identical plinks between source and destination pnodes.

<type> is the string identifier for the plink type.

Listing B.1 shows a truncated version of the .ptop text file, as the entire file is over 3,400 lines. Pnodes are either network switches or PC nodes in this .ptop file. Plinks are either gigabit ethernet links connecting PC nodes to PC network switches or ten-gigabit ethernet links connecting PC network switches to the master network switch. The truncated listing details all the switch nodes, all the interswitch links, two examples of the 525 PC node descriptions and three PC node link samples. Seven PC network switches, labeled "aswitch2" through "aswitch8", interconnect the testbed PC nodes. One master network switch, "aswitch1", connects the seven PC network switches together. All PC network switches connect to the master network switch by a single ten-gigabit ethernet link. Each PC node connects to its respective PC network switch by multiple gigabit ethernet links. PC nodes "apc2-1" and "apc2-6" are shown. Each of the PC nodes can host one "pc" or two "vm" vnodes. All the links of both PC nodes connect to PC network switch 2, labeled "aswitch2". The number of gigabit links per PC node and the network switches they connect to are listed in Table 3.1. Nomenclature such as "aswitch2" and "aswitch8" correspond to Network Switches 2 and 8 in Table 3.1. Similarly, "apc2-1" and "apc2-6" correspond to PC Nodes 1 and 6.

Listing B.1: Truncated .ptop File

```
node aswitch1 switch:1
node aswitch2 switch:1
node aswitch3 switch:1
node aswitch4 switch:1
node aswitch5 switch:1
node aswitch6 switch:1
node aswitch7 switch:1
node aswitch8 switch:1

link link-aswitch1:aswitch2 aswitch1 aswitch2 10000000 0 0 1 ethernet
link link-aswitch1:aswitch3 aswitch1 aswitch3 10000000 0 0 1 ethernet
link link-aswitch1:aswitch4 aswitch1 aswitch4 10000000 0 0 1 ethernet
link link-aswitch1:aswitch5 aswitch1 aswitch5 10000000 0 0 1 ethernet
link link-aswitch1:aswitch6 aswitch1 aswitch6 10000000 0 0 1 ethernet
link link-aswitch1:aswitch7 aswitch1 aswitch7 10000000 0 0 1 ethernet
link link-aswitch1:aswitch8 aswitch1 aswitch8 10000000 0 0 1 ethernet
```

```
node apc2-1 pc:1 delay:2 vm:2
link link-apc2-1:eth1-aswitch2:(null) apc2-1:apc1/eth1 aswitch2:(null
   ) 1000000 0 0 1 ethernet
link link-apc2-1:eth2-aswitch2:(null) apc2-1:apc1/eth2 aswitch2:(null
   ) 1000000 0 0 1 ethernet
.
.
link link-apc2-1:eth20-aswitch2:(null) apc2-1:apc1/eth20 aswitch2:(
   null) 1000000 0 0 1 ethernet


node apc2-6 pc:1 delay:2 vm:2
link link-apc2-6:eth1-aswitch2:(null) apc2-6:apc6/eth1 aswitch2:(null
   ) 1000000 0 0 1 ethernet
link link-apc2-6:eth2-aswitch2:(null) apc2-6:apc6/eth2 aswitch2:(null
   ) 1000000 0 0 1 ethernet
.
.
link link-apc2-6:eth15-aswitch2:(null) apc2-6:apc6/eth15 aswitch2:(
   null) 1000000 0 0 1 ethernet
```

B.2 SF Virtual Topologies (.top files)

Similar to the .ptop file, lines in the .top file describe vnodes or vlinks. Lines can also be used to identify fixed vnode assignments and vclasses. Lines describing vnodes are in the format:

```
node <node> <type> [<desires>]
```

<node> is the string identifier for the vnode.

<type> is the string identifier for the vnode type.

<desires> is a space-separated list of <desire>:<weight>.

<desire> is a string identifier of the desire.

<weight> is the cost of not having the desire fulfilled. A weight ≥ 1.0 will result in a violation if not fulfilled.

Lines describing vlinks are in the following format:

```
link <link> <src> <dst> <bw>[:<underbw>:<overbw>[:<weight>]]
<delay>[:<underdelay>:<overdelay>[:<weight>]]
<loss>[:<underloss>:<overloss>[:<weight>]]
<rbw>[:<underbw>:<overbw>[:<weight>]]
```

```
<rdelay>[:<underdelay>:<overdelay>[:<weight>]]
<rloss>[:<underloss>:<overloss>[:<weight>]]
<nodelay|mustdelay> <emulated> <type>
```

`<bw>`, `<delay>` and `<loss>` are the characteristics of the vlink. `<type>` is the string identifier for the vlink type. The reminder of the arguments are optional delta arguments that describe the range of error tolerance in the assignment (e.g., how far under and over the assignment can be by a given value). A value of 0 is the default and a value of -1 indicates that best effort is tolerable. The weights are optional floating point values that allow the user to specify the relative importance of the parameters. The default is 1. A user can also specify reverse delay characteristics. If these are omitted, the normal delay characteristics revert to the default. A `<nodelay>` value indicates the vlink should not be delayed. `<mustdelay>` indicates that vlink must be delayed.

Lines used to define fixed vnodes are in the format:

```
fix-node <node> <physical node>
```

Lines used to describe vclasses are in the format:

```
make-vclass <name> <weight> <physical types...>
```

There are multiple steps necessary to create SF and random virtual topologies in the format required by *assign*. The first step is to create either a BA SF or ER random network topology using BRITE. Table B.1 shows the BRITE parameters used to create both types of virtual topologies. A flat topology consisting of only one level is used for all virtual topologies, as hierarchical complex networks are beyond the scope of this research. BRITE governs the placement of the nodes in the topology using the “HS” and “LS” parameters. “HS” is the size of plane the nodes are placed on and “LS” is the number of squares that divide the “HS” plane. Both of these values remain at the default, since the number of nodes and their degree are the defining characteristics for SF and random networks, not node geographical location. “Random” is selected as the node placement parameter for the same reason. “N” is dependent on the virtual topology being created (e.g., “10” for the 10-node topology,

Table B.1: The BRITE parameters, their descriptions and values used to create the SF and random virtual topologies.

Parameter	**Description**	**SF**	**Random**
Topology Type	Flat or hierarchical topology	1	1
Level	Current level in hierarchy	AS ONLY	AS ONLY
HS	Size of 1 side of the plane	1000	1000
LS	Size of 1 side of high-level square	100	100
N	Number of nodes	*variable*	*variable*
Model	Model type	BA	Waxman
alpha	Waxman-specific exponent	*n/a*	0.15
beta	Waxman-specific exponent	*n/a*	0.2
Node Placement	How nodes are placed in the plane	Random	Random
m	Number of links per new node	2	2
Growth Type	How nodes join the topology	Incremental	Incremental
Pref. Conn	Preference for higher degree nodes	On	None
BWdist	Bandwidth assignment to links	Constant	Constant
MaxBW, MinBW	Link bandwidth	1500	1500

"50" for the 50-node topology, etc.) and two links per new node is selected to keep the number of links in the resulting topology to a reasonable amount.

As an example, Listing B.2 shows the BRITE output file for the 10-node SF virtual topology. The first half of the file is the node section. Node identifiers, cartesian coordinates, degree and other information is presented here. The first column, consisting of node identifiers, is the only information pertinent for the development of the .top file from the node section. The second part of the BRITE file is the edge section. Edge identifiers, source and destination nodes, Euclidean length, bandwidth and propagation delay is presented here. The first, second, third and sixth columns are necessary from this section to build the .top file. These columns are the edge identifiers, source node, destination node and bandwidth, respectively.

Listing B.2: BRITE Output File for the 10-node SF Virtual Topology

```
Topology: ( 10 Nodes , 17 Edges )
Model (4 - ASBarabasi):  10 1000 100 1  2  1 1500.0 1500.0

Nodes: ( 10 )
0       477     580     2       2       0       AS_NODE
```

```
1         381       659       2         2         1         AS_NODE
2         2         69        7         7         2         AS_NODE
3         930       184       3         3         3         AS_NODE
4         216       71        2         2         4         AS_NODE
5         591       113       2         2         5         AS_NODE
6         513       923       2         2         6         AS_NODE
7         379       683       4         4         7         AS_NODE
8         691       849       5         5         8         AS_NODE
9         270       417       5         5         9         AS_NODE


Edges: ( 17 )
0         2         4         214.00    0.71      1500.0   2    4    E_AS      U
1         2         9         439.23    1.46      1500.0   2    9    E_AS      U
2         4         9         350.18    1.16      1500.0   4    9    E_AS      U
3         8         2         1040.73   3.47      1500.0   8    2    E_AS      U
4         8         9         603.21    2.01      1500.0   8    9    E_AS      U
5         6         2         995.20    3.31      1500.0   6    2    E_AS      U
6         6         9         561.32    1.87      1500.0   6    9    E_AS      U
7         1         2         701.24    2.33      1500.0   1    2    E_AS      U
8         1         9         266.24    0.88      1500.0   1    9    E_AS      U
9         3         2         935.09    3.11      1500.0   3    2    E_AS      U
10        3         8         706.64    2.35      1500.0   3    8    E_AS      U
11        7         2         720.50    2.40      1500.0   7    2    E_AS      U
12        7         8         353.41    1.17      1500.0   7    8    E_AS      U
13        5         7         608.14    2.02      1500.0   5    7    E_AS      U
14        5         3         346.35    1.15      1500.0   5    3    E_AS      U
15        0         8         343.73    1.14      1500.0   0    8    E_AS      U
16        0         7         142.17    0.47      1500.0   0    7    E_AS      U
```

The vnodes created by the BRITE output files require type specification, either "pc" or "vm", the two types supported by the pnodes in the .ptop file. The "pc" type represents a vnode that consumes large amounts of computing resources and therefore can only be mapped one-to-one onto a given pnode. "vm" is a vnode type that requires substantially less computing resources, hence two can be multiplexed onto any single pnode. Table B.2 indicates which of the vnodes are specified as type "pc". The remainder of the vnodes in the .top files are specified as "vm". In topologies greater than ten nodes, "vm" vnodes outnumber "pc" vnodes to ensure a physical solution can be found with available testbed resources. The original bandwidth specification for all vlinks in the BRITE topologies is 1500 kilobits per second. A portion of these vlinks, listed in Table B.2, are increased to 100000 kilobits to represent link variation in actual workloads submitted to Emulab. In all virtual topologies, the

Table B.2: Vnode and vlink modifications made to SF virtual topologies produced by BRITE. The second column indicates which vnodes are specified as type "pc". The remaining vnodes in the .top file are designated as type "vm". The third column indicates how many vlinks are increased from 1500 to 10000 kilobits per second. The vlinks changed are the ones at the end of the .top files.

Virtual Topology	**Vnodes Specified as *pc***	**Total Number of Vlinks**	**Vlinks Increased to *100000***
10	0 through 4	17	last 8
20	0 through 9	37	last 17
30	0 through 9	57	last 27
40	0 through 9	77	last 37
50	0 through 9	97	last 47
60	0 through 9	117	last 47
70	0 through 9	137	last 47
80	0 through 9	157	last 47
90	0 through 9	177	last 47
100	0 through 9	197	last 47
200	0 through 9	397	last 47
300	0 through 9	597	last 47
400	0 through 9	797	last 47
500	0 through 9	997	last 47
600	0 through 9	1,197	last 47
700	0 through 9	1,397	last 47
800	0 through 9	1,597	last 47
900	0 through 9	1,797	last 47
1000	0 through 9	1,997	last 47

aggregate bandwidth of all vlinks is less than ten gigabits per second, ensuring a physical topology solution is possible even if all vlinks are mapped through the same inter-switch plink. This represents a worst-case scenario in vlink assignment and limited the number of vlinks that could be increased to 100000 kilobits per second in the 1000-node virtual topology. Additionally, the bandwidth for any single vlink is 1500 or 100000 kilobits per second, ensuring any one vlink cannot saturate a single PC node plink. All PC node plinks have a bandwidth of one gigabit per second.

Listing B.3 shows the entire .top text file for the 10-node SF virtual topology created using BRITE. Due to their length, the other .top files are not included, but

Table B.3: Vnode and vlink modifications made to random virtual topologies produced by BRITE. The second column indicates which vnodes are specified as type "pc". The remaining vnodes in the .top file are designated as type "vm". The third column indicates how many vlinks are increased from 1500 to 10000 kilobits per second. The vlinks changed are the ones at the end of the .top files.

Virtual Topology	**Vnodes Specified as *pc***	**Total Number of Vlinks**	**Vlinks Increased to *100000***
10	0 through 4	20	last 10
20	0 through 9	40	last 20
30	0 through 9	60	last 30
40	0 through 9	80	last 40
50	0 through 9	100	last 50
60	0 through 9	120	last 50
70	0 through 9	140	last 50
80	0 through 9	160	last 50
90	0 through 9	180	last 50
100	0 through 9	200	last 50
200	0 through 9	400	last 50
300	0 through 9	600	last 50
400	0 through 9	800	last 50
500	0 through 9	1,000	last 50
600	0 through 9	1,200	last 50
700	0 through 9	1,400	last 50
800	0 through 9	1,600	last 50
900	0 through 9	1,800	last 50
1000	0 through 9	2,000	last 50

can be recreated with the information provided in this section. The final modification to the .top file is the change in vlink identifier. The original BRITE numbering scheme is changed to an identifier composed of the source and destination nodes (e.g., vlink "0" is changed to vlink "linksimple/link2-4" where two is the source vnode and four is the destination vnode).

Listing B.3: Entire .top File for the 10-node SF Virtual Topology

```
node 0 pc
node 1 pc
node 2 pc
node 3 pc
node 4 pc
node 5 vm
```

```
node 6 vm
node 7 vm
node 8 vm
node 9 vm
link linksimple/link2-4 2 4 1500 0 0 ethernet
link linksimple/link2-9 2 9 1500 0 0 ethernet
link linksimple/link4-9 4 9 1500 0 0 ethernet
link linksimple/link8-2 8 2 1500 0 0 ethernet
link linksimple/link8-9 8 9 1500 0 0 ethernet
link linksimple/link6-2 6 2 1500 0 0 ethernet
link linksimple/link6-9 6 9 1500 0 0 ethernet
link linksimple/link1-2 1 2 1500 0 0 ethernet
link linksimple/link1-9 1 9 1500 0 0 ethernet
link linksimple/link3-2 3 2 100000 0 0 ethernet
link linksimple/link3-8 3 8 100000 0 0 ethernet
link linksimple/link7-2 7 2 100000 0 0 ethernet
link linksimple/link7-8 7 8 100000 0 0 ethernet
link linksimple/link5-7 5 7 100000 0 0 ethernet
link linksimple/link5-3 5 3 100000 0 0 ethernet
link linksimple/link0-8 0 8 100000 0 0 ethernet
link linksimple/link0-7 0 7 100000 0 0 ethernet
```

Appendix C. Data

This appendix contains the measured data from the first 20 trials of the experiments. Tables C.1 through C.10 show the execution time and objective function score results for both search algorithms when the 10 to 100-node random virtual topologies are mapped to the original set of testbed resources. Tables C.20 through C.29 show the results when the 10 to 100-node SF virtual topologies are mapped to the original set of testbed resources.

Tables C.11 through C.19 show the results when the 200 to 1000-node random virtual topologies (altered to pass *assign* prechecks) are mapped to the second set of testbed resources. Tables C.30 through C.38 show the results when the 200 to 1000-node SF virtual topologies are mapped to the second set of testbed resources. The second set of testbed resources is developed because no valid physical topology solution exists for the 200 to 1000-node random and SF virtual topologies using the original set of testbed resources and one-to-one link resolutions (see Section 4.2).

Table C.1: Measured data for the 10-node random virtual topology.

Search Algorithm	Violations	Score	Time	Total Iterations	Iterations to Best	Random Seed
SA	0	3.9	1.56844	36000	7184	127
SA	0	3.9	1.68774	39000	4814	254
SA	0	3.9	1.6019	37000	8299	381
SA	0	3.9	1.64446	38000	7622	508
SA	0	3.9	1.62702	38000	6848	635
SA	0	3.9	1.63375	38000	4082	762
SA	0	3.9	1.75976	40000	7705	889
SA	0	3.9	1.78881	42000	4402	1016
SA	0	3.9	1.69146	39000	8921	1143
SA	0	3.9	1.71644	39000	7182	1270
SA	0	3.9	2.0218	46000	6451	1397
SA	0	3.9	1.74369	40000	3383	1524
SA	0	3.9	1.76719	41000	7759	1651
SA	0	3.9	1.62583	38000	7020	1778
SA	0	3.9	1.61964	37000	8870	1905
SA	0	3.9	1.77459	41000	8510	2032
SA	0	3.4	1.78047	42000	35813	2159
SA	0	3.9	1.54453	36000	8357	2286
SA	0	3.9	1.80411	41000	2060	2413
SA	0	3.9	1.74001	40000	2034	2540
TS	0	3.9	0.315629	11395	6328	127
TS	0	4.4	0.321119	11681	684	254
TS	0	4.4	0.300928	11427	969	381
TS	0	4.4	0.308338	11535	5036	508
TS	0	3.9	0.319004	11723	1218	635
TS	0	4.4	0.318233	11935	759	762
TS	0	4.4	0.314141	11790	10758	889
TS	0	3.9	0.310395	11518	6725	1016
TS	0	4.4	0.324303	11850	84	1143
TS	0	3.9	0.312626	11552	175	1270
TS	0	4.4	0.322075	11755	4552	1397
TS	0	4.4	0.301234	11279	59	1524
TS	0	3.9	0.316211	11714	2313	1651
TS	0	4.4	0.316222	11384	7378	1778
TS	0	3.9	0.323637	11546	98	1905
TS	0	4.4	0.312331	11414	9383	2032
TS	0	4.4	0.319072	11566	195	2159
TS	0	3.9	0.312077	11513	1164	2286
TS	0	4.6	0.320933	11882	3992	2413
TS	0	4.4	0.324541	11881	2708	2540

Table C.2: Measured data for the 20-node random virtual topology.

Search Algorithm	Violations	Score	Time	Total Iterations	Iterations to Best	Random Seed
SA	0	6.1	2.15324	50000	41442	127
SA	0	6.6	1.81116	43000	14484	254
SA	0	6.6	1.698	40000	11812	381
SA	0	6.6	1.97506	46000	15118	508
SA	0	6.6	1.82943	43000	16541	635
SA	0	6.6	1.80088	42000	13279	762
SA	0	6.6	1.97695	46000	14688	889
SA	0	6.1	2.23101	51000	40784	1016
SA	0	6.6	1.40602	33000	15603	1143
SA	0	6.6	2.01113	47000	16620	1270
SA	0	6.1	2.02476	47000	36518	1397
SA	0	6.6	2.14878	50000	19390	1524
SA	0	6.1	2.05703	48000	35566	1651
SA	0	6.1	2.96138	67000	31169	1778
SA	0	6.6	2.36063	54000	11475	1905
SA	0	6.6	1.70081	40000	13300	2032
SA	0	6.6	1.48836	35000	10996	2159
SA	0	6.6	2.37055	55000	19036	2286
SA	0	6.6	2.14435	50000	15023	2413
SA	0	6.6	2.48568	58000	16571	2540
TS	0	13.3	0.531437	21874	15146	127
TS	0	11.2	0.530389	21777	21744	254
TS	0	9.94	0.53865	21892	6081	381
TS	0	9.44	0.537806	21976	9818	508
TS	0	6.8	0.533129	21913	16822	635
TS	0	12.72	0.533348	21828	11349	762
TS	0	9.26	0.533993	21816	17707	889
TS	0	10.4	0.55103	22282	19221	1016
TS	0	9.26	0.522243	21820	12711	1143
TS	0	9.06	0.541307	22185	3748	1270
TS	0	9.82	0.533998	21926	19851	1397
TS	0	10.2	0.529982	21778	15657	1524
TS	0	11.58	0.531412	21873	21546	1651
TS	0	11.16	0.52176	21357	2061	1778
TS	0	12.54	0.538586	22030	5800	1905
TS	0	9.06	0.538386	22016	13886	2032
TS	0	9.44	0.537849	22088	20020	2159
TS	0	9.06	0.541767	21876	12683	2286
TS	0	11.16	0.534206	21694	20660	2413
TS	0	6.8	0.52776	21887	18841	2540

Table C.3: Measured data for the 30-node random virtual topology.

Search Algorithm	Violations	Score	Time	Total Iterations	Iterations to Best	Random Seed
SA	0	8.4	2.0925	49000	21777	127
SA	0	8.4	1.55994	38000	18395	254
SA	0	8.4	1.88089	45000	18872	381
SA	0	8.4	2.13052	50000	17818	508
SA	0	8.4	2.13774	51000	19002	635
SA	0	8.4	1.81995	44000	19543	762
SA	0	8.4	2.05615	50000	21971	889
SA	0	8.4	1.87829	45000	18748	1016
SA	0	8.4	1.84629	44000	12197	1143
SA	0	8.4	1.46626	35000	14180	1270
SA	0	8.4	2.01955	48000	18522	1397
SA	0	8.4	2.17754	51000	19830	1524
SA	0	8.4	2.08118	49000	23449	1651
SA	0	8.4	1.93685	46000	25432	1778
SA	0	8.4	1.75991	42000	17293	1905
SA	0	8.4	2.07622	49000	25075	2032
SA	0	8.4	2.16359	51000	23720	2159
SA	0	7.9	2.244	52000	24205	2286
SA	0	8.4	2.24284	53000	19063	2413
SA	0	8.4	2.08482	49000	20935	2540
TS	0	13.84	0.785672	32683	12363	127
TS	0	11.06	0.771221	31990	26926	254
TS	0	16.06	0.76937	31804	28722	381
TS	0	17.44	0.771819	32109	2515	508
TS	0	16.22	0.776468	32146	32078	635
TS	0	13.2	0.763365	31924	31877	762
TS	0	12.5	0.761945	31809	31746	889
TS	0	14.22	0.764071	31858	30850	1016
TS	0	15.22	0.762834	32053	30991	1143
TS	0	13.76	0.762476	31924	31882	1270
TS	0	13.58	0.785012	32519	31482	1397
TS	0	16.92	0.752228	31661	30624	1524
TS	0	14.84	0.76694	32087	19841	1651
TS	0	14.36	0.770273	32018	26953	1778
TS	0	13.76	0.758866	31788	23672	1905
TS	0	13.58	0.772283	32347	13179	2032
TS	0	15.16	0.760435	31752	29699	2159
TS	0	16.42	0.782152	32050	24983	2286
TS	0	16.1	0.770897	32029	19929	2413
TS	0	16.3	0.763158	32014	30953	2540

Table C.4: Measured data for the 40-node random virtual topology.

Search Algorithm	Violations	Score	Time	Total Iterations	Iterations to Best	Random Seed
SA	0	10.2	2.22941	52000	21608	127
SA	0	10.2	2.13313	50000	22580	254
SA	0	10.2	2.23211	52000	22481	381
SA	0	10.2	1.97763	47000	17231	508
SA	0	10.2	1.97163	46000	21456	635
SA	0	10.2	1.18938	29000	24472	762
SA	0	10.2	1.85472	44000	24523	889
SA	0	10.2	2.22812	53000	27616	1016
SA	0	10.2	1.75657	42000	25436	1143
SA	0	10.2	1.811	43000	25427	1270
SA	0	10.2	2.00094	47000	20281	1397
SA	0	10.2	2.23606	52000	25972	1524
SA	0	10.2	1.89938	45000	18505	1651
SA	0	10.2	2.1818	51000	20118	1778
SA	0	10.2	1.85594	44000	25442	1905
SA	0	10.2	1.72778	41000	22965	2032
SA	0	10.2	1.8033	43000	28956	2159
SA	0	10.2	2.06377	48000	24436	2286
SA	0	10.2	2.08028	48000	24398	2413
SA	0	10.2	1.89801	45000	23121	2540
TS	0	21.12	1.02738	41834	18501	127
TS	0	20.7	1.02771	42073	39048	254
TS	0	23.06	1.02752	42032	15703	381
TS	0	22.72	1.02658	42527	42488	508
TS	0	20.36	1.02832	42044	42034	635
TS	0	19.66	1.0137	41901	36822	762
TS	0	20.32	1.02769	42256	37186	889
TS	0	21.76	1.01766	42114	39021	1016
TS	0	19.8	1.00618	41984	41953	1143
TS	0	20.9	1.02261	41938	36810	1270
TS	0	17.9	1.02185	41878	33769	1397
TS	0	21.88	1.0397	42318	35206	1524
TS	0	22.72	1.03912	42571	38539	1651
TS	0	22.14	1.0272	41925	27753	1778
TS	0	22.16	1.0341	42588	41583	1905
TS	0	21.02	1.02033	41806	41769	2032
TS	0	22.84	1.03898	42945	24818	2159
TS	0	22.28	1.01304	41930	24655	2286
TS	0	24.36	1.01213	41592	29475	2413
TS	0	26.68	1.01938	41807	40794	2540

Table C.5: Measured data for the 50-node random virtual topology.

Search Algorithm	Violations	Score	Time	Total Iterations	Iterations to Best	Random Seed
SA	0	12	1.74443	43000	36946	127
SA	0	12.2	1.32247	33000	24735	254
SA	0	15.22	1.75129	45000	22320	381
SA	0	12	1.61834	40000	27330	508
SA	0	16.42	1.16845	31000	21259	635
SA	0	12.2	1.33618	34000	32973	762
SA	0	12	1.81136	45000	28703	889
SA	0	17.94	0.983013	26000	21691	1016
SA	0	12.2	1.30335	33000	29282	1143
SA	0	12	2.00477	49000	35601	1270
SA	0	12.2	1.66512	43000	23634	1397
SA	0	14.46	1.10558	28000	21452	1524
SA	0	12	2.07901	50000	26890	1651
SA	0	12	1.38736	35000	27450	1778
SA	0	12	1.87182	47000	43095	1905
SA	0	12	1.51578	38000	33327	2032
SA	0	12	1.37811	35000	24601	2159
SA	0	12	2.41203	58000	30588	2286
SA	0	12.2	1.76681	44000	23593	2413
SA	0	12	1.58525	39000	24452	2540
TS	0	28.72	1.24255	52289	42171	127
TS	0	28.28	1.2474	52205	46140	254
TS	0	23.82	1.23974	52027	36856	381
TS	0	26.06	1.25267	52247	50200	508
TS	0	30.66	1.24936	52057	32855	635
TS	0	32.64	1.24916	52594	45475	762
TS	0	29.08	1.25084	52292	42150	889
TS	0	29.78	1.22621	51901	39846	1016
TS	0	23.36	1.23522	51899	31697	1143
TS	0	27.3	1.25128	52052	45992	1270
TS	0	28.36	1.24474	52211	47165	1397
TS	0	28.26	1.24062	52026	16700	1524
TS	0	30.46	1.25695	52045	31808	1651
TS	0	30.56	1.23893	52033	30792	1778
TS	0	26.04	1.25325	52555	50512	1905
TS	0	26.1	1.25345	52335	44222	2032
TS	0	25.16	1.25548	52619	36374	2159
TS	0	24.7	1.2337	52035	46946	2286
TS	0	25.72	1.24083	52131	43040	2413
TS	0	22.82	1.24256	52565	44454	2540

Table C.6: Measured data for the 60-node random virtual topology.

Search Algorithm	Violations	Score	Time	Total Iterations	Iterations to Best	Random Seed
SA	0	16.26	1.58316	40000	23816	127
SA	0	28.48	0.946807	25000	17553	254
SA	0	16.46	1.41667	36000	27195	381
SA	0	19.16	1.84582	48000	36086	508
SA	0	13.8	1.97871	49000	46682	635
SA	0	16.64	1.24574	32000	28242	762
SA	0	16.64	1.44258	37000	25228	889
SA	0	14	0.994978	26000	21920	1016
SA	0	13.8	1.77906	45000	33643	1143
SA	0	13.8	1.92122	47000	40709	1270
SA	0	14	1.43434	36000	27797	1397
SA	0	13.8	1.83981	46000	37259	1524
SA	0	13.8	1.79595	45000	34104	1651
SA	0	19.94	1.25175	32000	17168	1778
SA	0	26.76	0.91948	24000	22013	1905
SA	0	13.8	1.3709	35000	31024	2032
SA	0	13.8	1.53191	39000	34253	2159
SA	0	28.46	1.02124	27000	16448	2286
SA	0	13.8	1.84639	46000	34095	2413
SA	0	13.8	1.74185	44000	29333	2540
TS	0	34.02	1.47791	62006	61941	127
TS	0	32.16	1.48046	61841	56770	254
TS	0	36.4	1.48635	62101	56068	381
TS	0	30.34	1.49368	62401	37163	508
TS	0	32.4	1.49563	62099	53975	635
TS	0	40.18	1.47315	62117	34877	762
TS	0	31.14	1.48377	62448	31112	889
TS	0	32.6	1.4773	62118	60050	1016
TS	0	33.84	1.50088	62419	52340	1143
TS	0	36.18	1.47672	62547	52431	1270
TS	0	32.66	1.49036	62122	53019	1397
TS	0	34.86	1.49336	62218	61171	1524
TS	0	28.9	1.48183	62239	46057	1651
TS	0	31.58	1.49709	62592	61521	1778
TS	0	28.68	1.49749	62355	61308	1905
TS	0	34.32	1.48128	62248	43076	2032
TS	0	35.26	1.47516	62064	56029	2159
TS	0	35.06	1.48398	62083	60016	2286
TS	0	30.64	1.48092	62403	57297	2413
TS	0	34.58	1.48066	62152	59068	2540

Table C.7: Measured data for the 70-node random virtual topology.

Search Algorithm	Violations	Score	Time	Total Iterations	Iterations to Best	Random Seed
SA	0	28.06	1.47159	39000	32961	127
SA	0	25.58	1.21768	32000	28840	254
SA	1	16.36	0.916236	25000	23043	381
SA	0	15.6	1.82047	47000	33357	508
SA	0	33.02	1.35849	36000	20604	635
SA	0	22.62	1.35666	36000	32225	762
SA	0	40.78	1.12019	30000	20781	889
SA	0	22.36	1.01838	27000	25589	1016
SA	1	22.48	1.1945	32000	26272	1143
SA	1	35.42	0.691871	19000	17636	1270
SA	0	20.84	1.50316	39000	28598	1397
SA	0	27.06	1.41383	37000	24269	1524
SA	0	19.4	1.48732	39000	27736	1651
SA	1	26.14	0.784745	21000	18163	1778
SA	1	21.4	1.30595	35000	33560	1905
SA	0	15.6	1.77231	46000	43096	2032
SA	0	31.6	1.15954	32000	22436	2159
SA	1	26.04	0.920526	25000	19294	2286
SA	0	24.9	1.53938	41000	27424	2413
SA	0	25.08	1.7332	46000	28442	2540
TS	0	40.8	1.70228	72145	50969	127
TS	0	38.56	1.70554	72632	71559	254
TS	0	36.46	1.69686	72349	72295	381
TS	0	38.06	1.71238	72292	56130	508
TS	0	35.84	1.70108	72397	72319	635
TS	0	46.24	1.6949	72362	69348	762
TS	0	43.9	1.72665	72449	56395	889
TS	0	41.78	1.69921	72284	68251	1016
TS	0	38.38	1.70239	72181	72133	1143
TS	0	39.04	1.71361	72740	71724	1270
TS	0	39.02	1.70876	72235	36895	1397
TS	0	45.16	1.71006	72348	69279	1524
TS	0	43.44	1.69995	72226	67145	1651
TS	0	42.3	1.68784	71986	69960	1778
TS	0	38.7	1.71459	72793	59696	1905
TS	0	42.26	1.70382	72106	72086	2032
TS	0	35.96	1.69838	72471	55377	2159
TS	0	37.6	1.70823	72039	69967	2286
TS	0	39.78	1.69995	72758	68721	2413
TS	0	49.64	1.69831	72363	69324	2540

Table C.8: Measured data for the 80-node random virtual topology.

Search Algorithm	Violations	Score	Time	Total Iterations	Iterations to Best	Random Seed
SA	0	19.86	1.94566	47000	39029	127
SA	0	17.6	2.06504	50000	34027	254
SA	0	17.4	1.9932	48000	45913	381
SA	0	17.6	2.27296	54000	38063	508
SA	0	17.4	2.36893	56000	46020	635
SA	0	19.86	1.89973	46000	32023	762
SA	0	17.4	2.50716	58000	45219	889
SA	0	17.4	2.59908	61000	43715	1016
SA	0	17.4	2.02663	48000	38306	1143
SA	0	17.4	2.16053	52000	36910	1270
SA	0	17.4	1.99344	47000	45004	1397
SA	0	17.4	2.61277	61000	41425	1524
SA	0	17.6	1.82663	44000	42972	1651
SA	0	17.4	2.32419	55000	40389	1778
SA	0	17.4	2.15358	51000	33278	1905
SA	0	24.24	1.64099	40000	38979	2032
SA	0	17.4	2.46558	58000	40701	2159
SA	0	17.4	1.54542	38000	35979	2286
SA	0	17.4	2.21246	52000	36530	2413
SA	0	17.4	2.33284	56000	43056	2540
TS	0	49.74	1.93569	82191	81181	127
TS	0	48.3	1.92021	81937	76903	254
TS	0	48.42	1.92079	82256	78209	381
TS	0	49.72	1.93698	82379	52204	508
TS	0	47.96	1.93364	82380	52219	635
TS	0	51.52	1.9226	82363	52181	762
TS	1	46.02	1.94323	82904	74828	889
TS	0	48.3	1.92848	82317	66255	1016
TS	0	48.82	1.92749	82518	81494	1143
TS	0	51.72	1.9347	82504	80500	1270
TS	0	40.6	1.94246	82604	82542	1397
TS	0	46.08	1.92079	82141	66972	1524
TS	0	44.94	1.90885	82218	65113	1651
TS	0	43.62	1.94395	82672	67588	1778
TS	0	54.7	1.90653	82014	63898	1905
TS	0	45.18	1.92219	81989	76931	2032
TS	0	43.08	1.92831	82236	79179	2159
TS	0	51.66	1.92696	82294	66185	2286
TS	0	44.38	1.93321	82714	71661	2413
TS	0	47.12	1.93429	82144	70042	2540

Table C.9: Measured data for the 90-node random virtual topology.

Search Algorithm	Violations	Score	Time	Total Iterations	Iterations to Best	Random Seed
SA	2	37.9	0.930412	25920	23039	127
SA	2	34.22	1.64567	44280	25237	254
SA	3	39.52	1.08561	29160	27513	381
SA	1	30.52	1.22429	33480	27382	508
SA	3	43.38	1.12702	31320	19528	635
SA	0	27.92	1.51294	41040	32957	762
SA	2	44.9	0.847759	23760	17823	889
SA	3	48.9	1.04857	29160	20710	1016
SA	2	37.66	1.10992	30240	22327	1143
SA	1	35.3	1.27614	34560	29432	1270
SA	1	41.44	1.21178	33480	26181	1397
SA	2	41.12	1.22577	33480	25783	1524
SA	2	37.88	1.08432	30240	28163	1651
SA	2	50.8	1.16363	32400	20842	1778
SA	2	40.98	0.958171	25920	24180	1905
SA	1	35.38	1.60457	43200	29283	2032
SA	1	29.9	1.46043	38880	29844	2159
SA	1	36.64	1.42685	38880	37275	2286
SA	2	33.56	1.44576	39960	38018	2413
SA	2	55.24	1.13602	31320	25061	2540
TS	0	51	2.31737	99944	75988	127
TS	0	52.76	2.3437	99283	93847	254
TS	0	54.28	2.33746	99658	90958	381
TS	0	54.48	2.31807	99385	92877	508
TS	0	49.72	2.35862	99968	60822	635
TS	0	57.98	2.3264	99797	88924	762
TS	0	57.34	2.32206	99384	93953	889
TS	0	52.4	2.3325	99566	95205	1016
TS	0	55.18	2.32481	99490	88653	1143
TS	0	54.18	2.33622	99363	91737	1270
TS	0	51.74	2.33301	99754	96450	1397
TS	1	56.62	2.33357	99420	92906	1524
TS	0	53.92	2.32872	99603	98521	1651
TS	0	51.24	2.31187	99841	99826	1778
TS	0	55.78	2.31789	99264	94894	1905
TS	0	50.78	2.34465	99428	98301	2032
TS	0	53.16	2.3105	99545	68026	2159
TS	1	54.54	2.32622	99166	93720	2286
TS	0	54.84	2.33996	99190	99165	2413
TS	0	55.24	2.33584	99586	77872	2540

Table C.10: Measured data for the 100-node random virtual topology.

Search Algorithm	Violations	Score	Time	Total Iterations	Iterations to Best	Random Seed
SA	3	54.8	1.25373	34800	23202	127
SA	3	48.06	1.12932	31200	29077	254
SA	2	49	1.50756	40800	33707	381
SA	5	59.72	0.952909	26400	23172	508
SA	6	61.62	0.922998	26400	19491	635
SA	4	40.6	1.69928	46800	41024	762
SA	5	61.64	0.752772	21600	16794	889
SA	3	49.92	1.52017	42000	24337	1016
SA	1	49.2	1.08414	30000	20932	1143
SA	2	46.68	1.02347	28800	26282	1270
SA	2	47.2	1.3907	38400	33281	1397
SA	3	53.76	1.45623	39600	34584	1524
SA	4	52.38	1.24032	34800	29862	1651
SA	3	44.92	1.42988	39600	37662	1778
SA	6	48.44	1.10785	31200	26253	1905
SA	3	45.54	1.65743	45600	29850	2032
SA	3	49.94	1.12361	31200	27436	2159
SA	2	55.34	1.2769	34800	22944	2286
SA	7	57.6	0.838242	24000	20054	2413
SA	2	43.14	1.69799	45600	35972	2540
TS	1	67.12	2.8491	122127	119726	127
TS	1	65.34	2.84045	122467	114044	254
TS	1	69.5	2.86128	122495	118893	381
TS	0	66.36	2.8666	122595	122581	508
TS	1	63.28	2.81427	122044	120839	635
TS	0	58.12	2.84466	122116	112472	762
TS	1	64.9	2.85422	122801	113129	889
TS	0	64.32	2.82115	122364	121163	1016
TS	1	63.44	2.85057	122202	107751	1143
TS	1	60.86	2.85952	122916	120466	1270
TS	0	62.94	2.8465	122409	121197	1397
TS	0	57.04	2.84324	122428	118747	1524
TS	1	68.14	2.85096	122195	122183	1651
TS	1	63.4	2.83581	122199	111344	1778
TS	1	65.68	2.84133	122174	107722	1905
TS	2	61.36	2.82713	122727	113060	2032
TS	1	68.24	2.83259	122053	79941	2159
TS	1	65.68	2.81893	122185	93284	2286
TS	2	60.2	2.84806	122149	88433	2413
TS	0	66	2.84435	122379	118778	2540

Table C.11: Measured data for the 200-node random virtual topology.

Search Algorithm	Violations	Score	Time	Total Iterations	Iterations to Best	Random Seed
SA	0	38	0.060283	1000	775	127
SA	0	38.2	0.059768	1000	905	254
SA	0	38.2	0.059459	1000	626	381
SA	1	38.66	0.059963	1000	400	508
SA	0	38.2	0.061329	1000	419	635
SA	1	38.66	0.059463	1000	649	762
SA	0	38	0.059328	1000	200	889
SA	0	38.2	0.059391	1000	201	1016
SA	0	38.2	0.060106	1000	339	1143
SA	0	38.2	0.059017	1000	289	1270
SA	0	38.2	0.059984	1000	249	1397
SA	0	38.2	0.060218	1000	252	1524
SA	0	38.2	0.058667	1000	218	1651
SA	0	38	0.060212	1000	554	1778
SA	0	38.2	0.059307	1000	201	1905
SA	0	38	0.059652	1000	989	2032
SA	0	38.2	0.060037	1000	475	2159
SA	0	38.2	0.059229	1000	939	2286
SA	0	38.2	0.059207	1000	227	2413
SA	0	38.2	0.059456	1000	984	2540
TS	0	38	6.35722	205613	334	127
TS	0	38	6.24639	202454	1274	254
TS	0	38	6.32163	205523	268	381
TS	0	38	6.24611	202613	1072	508
TS	0	38	6.24326	202654	800	635
TS	0	38	6.29702	204501	1972	762
TS	0	38	6.30663	204575	200	889
TS	0	38	6.25391	202467	2820	1016
TS	0	38	6.22648	202289	728	1143
TS	0	38	6.25842	202631	213	1270
TS	0	38	6.26316	202461	2485	1397
TS	0	38	6.31219	205614	464	1524
TS	0	38	6.2888	202768	1245	1651
TS	0	38	6.24691	203525	2449	1778
TS	0	38	6.32546	205442	489	1905
TS	0	38	6.23778	202246	1886	2032
TS	0	38	6.35833	205615	403	2159
TS	0	38	6.32754	204598	436	2286
TS	0	38	6.32186	203588	300	2413
TS	0	38	6.24478	203061	2465	2540

Table C.12: Measured data for the 300-node random virtual topology.

Search Algorithm	Violations	Score	Time	Total Iterations	Iterations to Best	Random Seed
SA	0	56.2	0.074043	1000	631	127
SA	4	58.04	0.074687	1000	536	254
SA	2	57.12	0.074886	1000	365	381
SA	0	56	0.074971	1000	300	508
SA	1	56.66	0.075209	1000	804	635
SA	2	56.92	0.073405	1000	300	762
SA	1	56.66	0.075666	1000	564	889
SA	1	56.66	0.074832	1000	593	1016
SA	1	56.66	0.07422	1000	730	1143
SA	3	57.58	0.074034	1000	966	1270
SA	0	56	0.075205	1000	300	1397
SA	2	57.12	0.076021	1000	402	1524
SA	2	56.92	0.074564	1000	300	1651
SA	1	56.66	0.075437	1000	992	1778
SA	1	56.66	0.075012	1000	906	1905
SA	0	56	0.076381	1000	927	2032
SA	2	57.12	0.07553	1000	866	2159
SA	4	58.04	0.074579	1000	554	2286
SA	0	56.2	0.074419	1000	322	2413
SA	0	56	0.074862	1000	420	2540
TS	0	56	10.8622	305982	2137	127
TS	0	56	10.9204	303467	1979	254
TS	0	56	10.7742	303837	1885	381
TS	0	56	10.7829	303965	300	508
TS	0	56	10.8385	304011	1544	635
TS	0	56	10.7443	303842	475	762
TS	0	56	10.8795	304589	2251	889
TS	0	56	10.8859	304138	1661	1016
TS	0	56	10.8401	303549	2913	1143
TS	0	56	10.8707	305061	1037	1270
TS	0	56	10.7999	306071	300	1397
TS	0	56	10.9036	306940	7621	1524
TS	0	56	10.8959	303514	3368	1651
TS	0	56	10.7512	303907	4080	1778
TS	0	56	10.8474	307025	8227	1905
TS	0	56	10.8886	306063	2087	2032
TS	0	56	10.7846	304010	2113	2159
TS	0	56	10.8464	305918	8252	2286
TS	0	56	10.8281	304069	429	2413
TS	0	56	10.9215	305981	7580	2540

Table C.13: Measured data for the 400-node random virtual topology.

Search Algorithm	Violations	Score	Time	Total Iterations	Iterations to Best	Random Seed
SA	2	75.12	0.092939	1000	740	127
SA	1	74.46	0.09295	1000	400	254
SA	3	75.58	0.09452	1000	524	381
SA	2	74.92	0.094024	1000	695	508
SA	1	74.66	0.093599	1000	584	635
SA	1	74.46	0.093848	1000	679	762
SA	3	75.58	0.094454	1000	823	889
SA	3	75.58	0.093492	1000	410	1016
SA	2	75.12	0.093147	1000	565	1143
SA	0	74.2	0.094446	1000	662	1270
SA	6	76.96	0.093365	1000	404	1397
SA	1	74.66	0.094049	1000	995	1524
SA	1	74.46	0.092782	1000	400	1651
SA	3	75.58	0.094427	1000	994	1778
SA	6	76.96	0.093195	1000	475	1905
SA	2	75.12	0.093537	1000	668	2032
SA	1	74.66	0.094303	1000	523	2159
SA	7	77.22	0.094401	1000	400	2286
SA	4	76.04	0.094154	1000	975	2413
SA	1	74.66	0.092731	1000	792	2540
TS	0	74	16.3862	407234	11819	127
TS	0	74	16.2251	404856	10045	254
TS	0	74	16.2914	408187	14794	381
TS	0	74	16.276	405355	2771	508
TS	0	74	16.3349	404607	4971	635
TS	0	74	16.3656	407243	4144	762
TS	0	74	16.2676	405400	4847	889
TS	0	74	16.3104	408251	16152	1016
TS	0	74	16.225	404627	1604	1143
TS	0	74	16.1714	405108	3039	1270
TS	0	74	16.3225	406166	3861	1397
TS	0	74	16.3003	405115	9727	1524
TS	0	74	16.3272	408242	976	1651
TS	0	74	16.1768	403828	2887	1778
TS	0	74	16.1891	403936	2754	1905
TS	0	74	16.2964	406275	8902	2032
TS	0	74	16.2637	405483	1293	2159
TS	0	74	16.2194	403368	4212	2286
TS	0	74	16.3184	405228	8678	2413
TS	0	74	16.354	405173	2435	2540

Table C.14: Measured data for the 500-node random virtual topology.

Search Algorithm	Violations	Score	Time	Total Iterations	Iterations to Best	Random Seed
SA	6	94.96	0.116826	1000	909	127
SA	9	96.34	0.118661	1000	980	254
SA	4	94.04	0.116565	1000	835	381
SA	4	93.84	0.115747	1000	500	508
SA	5	94.5	0.116753	1000	583	635
SA	6	94.96	0.117529	1000	962	762
SA	11	97.26	0.116763	1000	607	889
SA	4	93.84	0.116336	1000	500	1016
SA	4	94.04	0.11681	1000	546	1143
SA	6	94.96	0.116341	1000	950	1270
SA	11	97.26	0.119628	1000	888	1397
SA	5	94.3	0.116269	1000	500	1524
SA	4	93.84	0.115653	1000	500	1651
SA	3	93.58	0.116595	1000	597	1778
SA	7	95.42	0.116235	1000	618	1905
SA	6	94.96	0.11713	1000	599	2032
SA	8	95.68	0.115257	1000	500	2159
SA	7	95.42	0.118147	1000	762	2286
SA	5	94.5	0.118677	1000	930	2413
SA	5	94.5	0.116187	1000	658	2540
TS	0	92.2	22.2957	506635	6934	127
TS	0	92.2	22.2277	504738	127402	254
TS	0	92.2	22.3297	505522	59045	381
TS	0	92.2	22.3603	509365	12759	508
TS	0	92.2	22.3898	505999	33404	635
TS	0	92.2	22.3571	508288	34834	762
TS	0	92.2	22.4939	509507	73042	889
TS	0	92.2	22.2546	504311	95877	1016
TS	0	92.2	22.4726	508381	21802	1143
TS	0	92.2	22.3499	505792	55344	1270
TS	0	92.2	22.2555	505210	59753	1397
TS	0	92.2	22.3086	505923	34446	1524
TS	0	92.2	22.3661	505961	66530	1651
TS	0	92.2	22.2963	505365	9653	1778
TS	0	92.2	22.385	507256	69790	1905
TS	0	92.2	22.3742	509487	24943	2032
TS	0	92.2	22.4788	506477	16891	2159
TS	0	92.2	22.2521	505463	13835	2286
TS	0	92.2	22.3019	506244	42557	2413
TS	0	92.2	22.269	504756	35221	2540

Table C.15: Measured data for the 600-node random virtual topology.

Search Algorithm	Violations	Score	Time	Total Iterations	Iterations to Best	Random Seed
SA	2	111.12	0.144297	1000	973	127
SA	6	112.96	0.144437	1000	610	254
SA	0	110	0.14544	1000	634	381
SA	9	114.34	0.145495	1000	707	508
SA	2	110.92	0.144784	1000	757	635
SA	2	111.12	0.144109	1000	811	762
SA	8	113.88	0.146355	1000	897	889
SA	6	112.76	0.144849	1000	600	1016
SA	1	110.46	0.143836	1000	600	1143
SA	8	113.68	0.14351	1000	600	1270
SA	3	111.58	0.145469	1000	796	1397
SA	2	110.92	0.143868	1000	600	1524
SA	0	110	0.144742	1000	600	1651
SA	3	111.38	0.14436	1000	600	1778
SA	3	111.58	0.144932	1000	733	1905
SA	1	110.46	0.14401	1000	600	2032
SA	0	110	0.145047	1000	766	2159
SA	13	116.18	0.143891	1000	627	2286
SA	0	110.2	0.144863	1000	786	2413
SA	3	111.58	0.144488	1000	833	2540
TS	0	110	29.1294	605444	8869	127
TS	0	110	29.1246	606206	4870	254
TS	0	110	29.1925	606022	8052	381
TS	0	110	29.3888	607288	5060	508
TS	0	110	29.1106	606025	5361	635
TS	0	110	29.1972	605692	4403	762
TS	0	110	29.1852	605825	6688	889
TS	0	110	29.1716	605820	9201	1016
TS	0	110	29.2877	609349	3331	1143
TS	0	110	29.2103	606444	10770	1270
TS	0	110	29.2205	606440	3901	1397
TS	0	110	29.1667	607427	5599	1524
TS	0	110	29.046	604608	600	1651
TS	0	110	29.2008	606070	3915	1778
TS	0	110	29.2698	608325	3827	1905
TS	0	110	29.1399	606256	1379	2032
TS	0	110	29.0269	604967	3174	2159
TS	0	110	29.1536	606978	11867	2286
TS	0	110	29.1767	605879	6061	2413
TS	0	110	29.1932	605787	2819	2540

Table C.16: Measured data for the 700-node random virtual topology.

Search Algorithm	Violations	Score	Time	Total Iterations	Iterations to Best	Random Seed
SA	1	128.66	0.175748	1000	895	127
SA	4	129.84	0.177216	1000	700	254
SA	6	130.96	0.177854	1000	830	381
SA	2	128.92	0.175217	1000	700	508
SA	2	128.92	0.175825	1000	700	635
SA	10	132.6	0.174158	1000	700	762
SA	2	129.12	0.177823	1000	744	889
SA	4	129.84	0.176915	1000	700	1016
SA	5	130.5	0.178011	1000	805	1143
SA	6	130.76	0.174825	1000	700	1270
SA	1	128.46	0.178544	1000	911	1397
SA	2	129.12	0.178288	1000	828	1524
SA	2	129.12	0.176231	1000	780	1651
SA	3	129.38	0.177125	1000	700	1778
SA	6	130.76	0.175539	1000	700	1905
SA	4	129.84	0.176969	1000	700	2032
SA	6	130.96	0.177322	1000	708	2159
SA	0	128.2	0.178206	1000	704	2286
SA	4	130.04	0.17735	1000	982	2413
SA	0	128.2	0.176819	1000	719	2540
TS	0	128	36.6524	706238	6780	127
TS	0	128	36.5772	706662	7145	254
TS	0	128	36.5567	705335	19910	381
TS	0	128	36.6927	707115	5474	508
TS	0	128	36.5788	706202	4519	635
TS	0	128	36.7086	706245	7065	762
TS	0	128	36.5803	706437	3241	889
TS	0	128	36.5681	706121	9413	1016
TS	0	128	36.5612	706731	16874	1143
TS	0	128	36.6434	707007	14033	1270
TS	0	128	36.5602	706870	13526	1397
TS	0	128	36.6157	707418	18432	1524
TS	0	128	36.5709	706281	6087	1651
TS	0	128	36.657	708231	4890	1778
TS	0	128	36.7381	707359	7429	1905
TS	0	128	36.6449	707111	5541	2032
TS	0	128	36.5377	705329	2706	2159
TS	0	128	36.599	706777	3682	2286
TS	0	128	36.5992	706200	6884	2413
TS	0	128	36.6454	707078	9240	2540

Table C.17: Measured data for the 800-node random virtual topology.

Search Algorithm	Violations	Score	Time	Total Iterations	Iterations to Best	Random Seed
SA	9	150.14	0.21419	1000	800	127
SA	4	147.84	0.214007	1000	800	254
SA	5	148.3	0.215668	1000	800	381
SA	9	150.34	0.215337	1000	927	508
SA	5	148.5	0.21626	1000	890	635
SA	5	148.5	0.216182	1000	957	762
SA	12	151.52	0.214648	1000	800	889
SA	6	148.96	0.21592	1000	825	1016
SA	8	149.88	0.212846	1000	830	1143
SA	10	150.8	0.215858	1000	920	1270
SA	8	149.68	0.215758	1000	800	1397
SA	8	149.68	0.213578	1000	800	1524
SA	3	147.58	0.215846	1000	956	1651
SA	10	150.8	0.216997	1000	839	1778
SA	5	148.5	0.213762	1000	831	1905
SA	5	148.5	0.214052	1000	958	2032
SA	5	148.3	0.214286	1000	800	2159
SA	14	152.64	0.213204	1000	964	2286
SA	6	148.96	0.215908	1000	806	2413
SA	12	151.52	0.215296	1000	800	2540
TS	0	146.2	44.7326	805705	293195	127
TS	0	146.2	44.758	807014	341582	254
TS	0	146.2	44.6575	807079	323924	381
TS	0	146.2	44.7888	806444	8375	508
TS	0	146.2	44.8306	807162	68420	635
TS	0	146.2	44.9243	805523	2224	762
TS	0	146.2	44.7039	806716	327244	889
TS	0	146.2	44.8484	807291	172023	1016
TS	0	146.2	44.5919	807127	27341	1143
TS	0	146.2	44.6075	807690	62518	1270
TS	0	146.2	44.5866	806202	144548	1397
TS	0	146.2	44.7466	807715	125025	1524
TS	0	146.2	44.7023	806105	229568	1651
TS	0	146.2	44.8195	810089	80339	1778
TS	0	146.2	44.5909	805549	227987	1905
TS	0	146.2	44.6546	806637	30518	2032
TS	0	146.2	44.732	806871	77016	2159
TS	0	146.2	44.6824	807176	141252	2286
TS	0	146.2	44.7054	806896	13081	2413
TS	0	146.2	44.7544	808040	12558	2540

Table C.18: Measured data for the 900-node random virtual topology.

Search Algorithm	Violations	Score	Time	Total Iterations	Iterations to Best	Random Seed
SA	3	165.38	0.256272	1000	900	127
SA	5	166.3	0.25407	1000	900	254
SA	1	164.46	0.256863	1000	900	381
SA	9	168.14	0.254625	1000	900	508
SA	10	168.6	0.254332	1000	900	635
SA	4	166.04	0.256587	1000	982	762
SA	25	175.5	0.255208	1000	900	889
SA	1	164.46	0.255799	1000	900	1016
SA	3	165.38	0.256142	1000	900	1143
SA	3	165.38	0.256775	1000	900	1270
SA	4	166.04	0.258114	1000	949	1397
SA	6	166.76	0.255871	1000	900	1524
SA	10	168.6	0.253049	1000	900	1651
SA	6	166.76	0.257539	1000	900	1778
SA	3	165.38	0.255536	1000	900	1905
SA	4	165.84	0.255442	1000	900	2032
SA	4	165.84	0.255993	1000	900	2159
SA	6	166.76	0.256277	1000	900	2286
SA	4	165.84	0.25792	1000	900	2413
SA	5	166.3	0.256713	1000	900	2540
TS	0	164	53.5288	906962	5652	127
TS	0	164	53.5001	908102	5055	254
TS	0	164	53.4929	906968	8935	381
TS	0	164	53.6092	906231	8743	508
TS	0	164	53.4527	905885	4307	635
TS	0	164	53.4753	906996	12130	762
TS	0	164	53.553	907234	4651	889
TS	0	164	53.4876	906453	11048	1016
TS	0	164	53.4893	906399	8298	1143
TS	0	164	53.4947	905945	7232	1270
TS	0	164	53.4648	906374	6459	1397
TS	0	164	53.5758	907925	1787	1524
TS	0	164	53.4941	906586	12693	1651
TS	0	164	53.515	907498	5543	1778
TS	0	164	53.4877	906451	16380	1905
TS	0	164	53.4907	907319	9897	2032
TS	0	164	53.5148	908129	6527	2159
TS	0	164	53.4629	907386	9603	2286
TS	0	164	53.5973	909245	5614	2413
TS	0	164	53.4737	907118	13482	2540

Table C.19: Measured data for the 1000-node random virtual topology.

Search Algorithm	Violations	Score	Time	Total Iterations	Iterations to Best	Random Seed
SA	6	184.76	0.302967	1000	1000	127
SA	15	188.9	0.303758	1000	1000	254
SA	16	189.36	0.303676	1000	1000	381
SA	13	187.98	0.304592	1000	1000	508
SA	5	184.3	0.303673	1000	1000	635
SA	1	182.46	0.304128	1000	1000	762
SA	8	185.68	0.302922	1000	1000	889
SA	11	187.06	0.302362	1000	1000	1016
SA	11	187.06	0.302157	1000	1000	1143
SA	5	184.3	0.305702	1000	1000	1270
SA	15	188.9	0.303079	1000	1000	1397
SA	4	183.84	0.30146	1000	1000	1524
SA	17	189.82	0.300889	1000	1000	1651
SA	11	187.06	0.30411	1000	1000	1778
SA	5	184.3	0.305911	1000	1000	1905
SA	13	187.98	0.303527	1000	1000	2032
SA	8	185.68	0.30444	1000	1000	2159
SA	9	186.14	0.301071	1000	1000	2286
SA	12	187.52	0.302551	1000	1000	2413
SA	7	185.22	0.303279	1000	1000	2540
TS	0	182	63.4574	1007772	16870	127
TS	0	182	63.4664	1006887	29270	254
TS	0	182	63.4442	1006677	7962	381
TS	0	182	63.5178	1007202	10875	508
TS	0	182	63.499	1008163	23218	635
TS	0	182	63.4164	1006280	8005	762
TS	0	182	63.2501	1005739	2789	889
TS	0	182	63.4171	1006763	9289	1016
TS	0	182	63.3544	1006811	13784	1143
TS	0	182	63.5395	1007456	21471	1270
TS	0	182	63.5005	1007743	5131	1397
TS	0	182	63.4579	1007489	11074	1524
TS	0	182	63.4897	1006693	10327	1651
TS	0	182	63.6402	1007825	19278	1778
TS	0	182	63.4258	1006531	10263	1905
TS	0	182	63.4993	1006803	5781	2032
TS	0	182	63.5371	1007535	17170	2159
TS	0	182	63.3478	1006518	28653	2286
TS	0	182	63.4114	1006397	11297	2413
TS	0	182	63.4076	1008267	13350	2540

Table C.20: Measured data for the 10-node scale-free virtual topology.

Search Algorithm	Violations	Score	Time	Total Iterations	Iterations to Best	Random Seed
SA	0	3.78	1.40588	37000	8605	127
SA	0	3.78	1.14183	30000	5238	254
SA	0	3.78	1.42043	37000	6399	381
SA	0	3.78	1.04858	28000	3605	508
SA	0	3.78	1.61321	42000	5150	635
SA	0	3.28	1.11778	30000	17083	762
SA	0	3.78	1.32259	35000	5566	889
SA	0	3.78	1.36515	36000	5530	1016
SA	0	3.78	1.33308	36000	5666	1143
SA	0	3.78	1.31842	35000	4441	1270
SA	0	3.78	1.39767	37000	6089	1397
SA	0	3.78	1.36253	36000	8107	1524
SA	0	3.78	1.58718	42000	3680	1651
SA	0	3.78	1.16126	31000	8457	1778
SA	0	3.78	1.32347	35000	5353	1905
SA	0	3.78	1.38344	36000	7580	2032
SA	0	3.78	1.24381	33000	8048	2159
SA	0	3.78	1.24988	33000	8755	2286
SA	0	3.78	1.49776	39000	5442	2413
SA	0	3.78	1.32514	35000	7862	2540
TS	0	3.78	0.26538	11418	1035	127
TS	0	3.78	0.27916	11729	365	254
TS	0	4.28	0.255718	11351	960	381
TS	0	4.28	0.264208	11443	376	508
TS	0	4.48	0.26988	11689	913	635
TS	0	4.28	0.264067	11585	9641	762
TS	0	3.78	0.260931	11380	4230	889
TS	0	3.78	0.268701	11574	8468	1016
TS	0	4.28	0.263	11533	7768	1143
TS	0	4.28	0.265285	11745	2474	1270
TS	0	3.98	0.259769	11466	1231	1397
TS	0	4.48	0.276378	11446	1612	1524
TS	0	6.04	0.263257	11435	10459	1651
TS	0	3.98	0.264582	11647	9619	1778
TS	0	3.78	0.266402	11255	9729	1905
TS	0	3.78	0.266456	11352	1682	2032
TS	0	4.28	0.265807	11269	8164	2159
TS	0	4.28	0.263714	11494	6679	2286
TS	0	3.98	0.269749	11593	6457	2413
TS	0	3.78	0.275366	11883	2687	2540

Table C.21: Measured data for the 20-node scale-free virtual topology.

Search Algorithm	Violations	Score	Time	Total Iterations	Iterations to Best	Random Seed
SA	0	6.48	1.88283	47000	14783	127
SA	0	6.48	1.65063	41000	15549	254
SA	0	6.48	1.5969	40000	14824	381
SA	0	6.48	1.84905	46000	14261	508
SA	0	5.98	1.86693	46000	12485	635
SA	0	5.98	1.51489	38000	24579	762
SA	0	6.48	2.03753	51000	13323	889
SA	0	6.48	2.0866	52000	13320	1016
SA	0	6.48	1.46173	37000	13763	1143
SA	0	6.48	1.7856	44000	13420	1270
SA	0	5.98	1.51262	38000	33598	1397
SA	0	6.18	1.9578	49000	28922	1524
SA	0	6.48	1.42469	36000	17644	1651
SA	0	5.98	1.61566	40000	22786	1778
SA	0	6.48	1.80081	45000	14687	1905
SA	0	6.48	1.50116	38000	15194	2032
SA	0	6.48	1.60988	41000	12978	2159
SA	0	6.48	1.61717	41000	12418	2286
SA	0	5.98	1.90029	47000	16155	2413
SA	0	6.18	1.49799	38000	29170	2540
TS	0	9.72	0.524453	21594	6383	127
TS	0	9.9	0.536233	22243	12122	254
TS	0	10.08	0.518159	21514	3108	381
TS	0	6.68	0.522259	21626	19584	508
TS	0	9.32	0.548484	22228	4881	635
TS	0	9.7	0.526793	21679	8389	762
TS	0	9.7	0.524514	21502	20587	889
TS	0	8.94	0.521549	21512	10375	1016
TS	0	10.28	0.522781	21466	2086	1143
TS	0	8.94	0.523677	21686	18651	1270
TS	0	8.94	0.520772	21675	6304	1397
TS	0	10.9	0.538178	22130	22102	1524
TS	0	9.12	0.524878	21585	13386	1651
TS	0	6.68	0.536687	22059	22030	1778
TS	0	10.46	0.520909	21454	20271	1905
TS	0	10.46	0.541462	22214	16523	2032
TS	0	10.58	0.531712	21779	4642	2159
TS	0	10.9	0.524045	21779	1190	2286
TS	0	6.88	0.530925	21783	7370	2413
TS	0	10.28	0.527689	21740	14598	2540

Table C.22: Measured data for the 30-node scale-free virtual topology.

Search Algorithm	Violations	Score	Time	Total Iterations	Iterations to Best	Random Seed
SA	0	11.7	0.588505	21000	13832	127
SA	0	8.48	0.61856	18000	16279	254
SA	0	22.26	0.370239	17000	6423	381
SA	0	20.72	0.429969	19000	7716	508
SA	0	14.14	0.540104	17000	14975	635
SA	0	8.48	0.852883	29000	17904	762
SA	0	15.62	0.525677	17000	14925	889
SA	0	8.48	0.965421	28000	21771	1016
SA	0	15.56	0.503705	19000	17420	1143
SA	0	8.48	1.09187	36000	23499	1270
SA	0	15.68	0.552267	17000	8361	1397
SA	0	8.48	0.75025	25000	16145	1524
SA	0	15.98	0.461074	17000	10644	1651
SA	0	11.88	0.552655	19000	12291	1778
SA	0	8.28	0.706546	21000	18588	1905
SA	0	8.48	0.683239	23000	14622	2032
SA	0	19.84	0.503492	17000	7470	2159
SA	0	8.48	0.744015	22000	17680	2286
SA	0	19.2	0.495956	19000	10648	2413
SA	0	17.82	0.466762	17000	8607	2540
TS	0	16.36	0.787928	31814	4444	127
TS	0	16.56	0.747613	31890	25805	254
TS	0	16.36	0.782099	31986	29915	381
TS	0	12.96	0.708411	31926	23843	508
TS	0	17.32	0.759936	31927	30871	635
TS	0	15.44	0.782952	31921	21740	762
TS	0	15.42	0.772752	32013	25884	889
TS	0	15.22	0.803539	31925	9633	1016
TS	0	13.64	0.773243	32220	5736	1143
TS	0	16.3	0.789924	31961	21798	1270
TS	0	18.08	0.733487	32305	11866	1397
TS	0	15.56	0.776776	32069	30019	1524
TS	0	17.06	0.738385	31664	28606	1651
TS	0	11.7	0.723613	31756	22654	1778
TS	0	16.18	0.731286	31978	25803	1905
TS	0	16.56	0.788602	31938	30922	2032
TS	0	17.06	0.753653	31822	25739	2159
TS	0	10.74	0.783503	32368	12118	2286
TS	0	19.08	0.775243	31751	4427	2413
TS	0	12.5	0.738419	31869	29759	2540

Table C.23: Measured data for the 40-node scale-free virtual topology.

Search Algorithm	Violations	Score	Time	Total Iterations	Iterations to Best	Random Seed
SA	0	10.08	1.40472	35000	20819	127
SA	0	10.08	1.40108	35000	22194	254
SA	0	10.08	1.37056	34000	24045	381
SA	0	10.08	1.65836	42000	17898	508
SA	0	10.08	1.56357	39000	27086	635
SA	0	10.08	1.60305	40000	22762	762
SA	0	10.08	1.56073	39000	19075	889
SA	0	10.08	1.35124	34000	27468	1016
SA	0	10.08	1.37771	35000	21222	1143
SA	0	10.08	2.05633	50000	19063	1270
SA	0	10.08	1.4307	37000	27200	1397
SA	0	10.08	1.50085	37000	22275	1524
SA	0	10.08	1.50397	38000	21535	1651
SA	0	10.08	1.50003	37000	20614	1778
SA	0	10.08	1.81037	45000	26846	1905
SA	0	10.08	1.62775	40000	23820	2032
SA	0	10.08	1.22013	31000	19092	2159
SA	0	10.08	1.58326	40000	21907	2286
SA	0	10.08	1.71025	43000	26118	2413
SA	0	10.08	1.03567	27000	22207	2540
TS	0	21.92	1.11904	42318	20990	127
TS	0	20.7	1.11318	42081	24888	254
TS	0	19.04	1.10234	42119	31991	381
TS	0	18.36	1.1033	42281	42268	508
TS	0	24.56	1.09073	41765	32700	635
TS	0	21.08	1.09155	42052	26839	762
TS	0	18.94	1.0959	41879	26718	889
TS	0	18.8	1.1029	42423	38298	1016
TS	0	19.76	1.09343	42043	42004	1143
TS	0	17.42	1.10518	42385	26220	1270
TS	0	22.8	1.08746	41762	18509	1397
TS	0	18.98	1.09633	41872	39853	1524
TS	0	22.42	1.10216	42111	21803	1651
TS	0	19.24	1.09678	41805	18537	1778
TS	0	17.84	1.09124	41815	29622	1905
TS	0	19.36	1.10216	42128	36052	2032
TS	0	19.12	1.10986	42209	38152	2159
TS	0	20.8	1.10131	41944	33833	2286
TS	0	18.16	1.09171	41880	41839	2413
TS	0	21	1.1154	42558	24226	2540

Table C.24: Measured data for the 50-node scale-free virtual topology.

Search Algorithm	Violations	Score	Time	Total Iterations	Iterations to Best	Random Seed
SA	1	27.06	0.522652	17000	14826	127
SA	1	29.2	0.54723	17000	12289	254
SA	0	26.48	0.546879	17000	15779	381
SA	0	42.24	0.520343	17000	7905	508
SA	0	21.6	1.23594	40000	34556	635
SA	2	34.84	0.323601	17000	8496	762
SA	0	21.1	0.639378	19000	15635	889
SA	1	33.18	0.549402	20000	7315	1016
SA	0	40.3	0.488465	17000	7834	1143
SA	0	28.1	0.808628	25000	11143	1270
SA	0	30.86	0.643045	20000	17937	1397
SA	1	33.24	0.424706	17000	7851	1524
SA	0	24.22	0.693397	21000	19474	1651
SA	0	33.14	0.516222	17000	13685	1778
SA	1	27.26	0.504011	17000	10964	1905
SA	1	37.16	0.435947	17000	5641	2032
SA	1	29.78	0.481167	17000	11369	2159
SA	1	31.76	0.474848	17000	10169	2286
SA	0	33.88	0.58892	17000	11509	2413
SA	1	24.44	0.534299	18000	14401	2540
TS	0	27.88	1.18124	52014	50970	127
TS	0	30.16	1.20146	52615	35455	254
TS	0	27.94	1.17712	52208	20871	381
TS	0	23.58	1.17768	51898	48854	508
TS	0	27.84	1.18587	52155	20789	635
TS	0	29.52	1.16216	52414	52372	762
TS	0	26.34	1.16834	51869	44761	889
TS	0	25.84	1.17249	52449	52408	1016
TS	0	27	1.18588	52597	34447	1143
TS	0	27.98	1.17356	52254	44160	1270
TS	0	25.98	1.19156	52272	25945	1397
TS	0	29.92	1.1912	52070	51023	1524
TS	0	30.62	1.2084	52969	24711	1651
TS	0	30.74	1.19156	51949	50935	1778
TS	0	28.14	1.15119	52275	52267	1905
TS	0	23.58	1.1872	52038	50986	2032
TS	0	24.58	1.15444	52357	46261	2159
TS	0	29.66	1.18928	52146	41069	2286
TS	0	27.2	1.09357	52417	50406	2413
TS	0	25.34	1.15614	52073	51014	2540

Table C.25: Measured data for the 60-node scale-free virtual topology.

Search Algorithm	Violations	Score	Time	Total Iterations	Iterations to Best	Random Seed
SA	1	41.4	0.568395	17000	10582	127
SA	1	41.86	0.543341	18000	14364	254
SA	1	31.26	0.946329	31000	19878	381
SA	2	36.28	0.551191	20000	18169	508
SA	1	24.86	0.613986	19000	17645	635
SA	2	45.08	0.434839	17000	9839	762
SA	2	38.82	0.517031	17000	11974	889
SA	0	35.38	0.551866	17000	13413	1016
SA	1	49.76	0.582756	17000	8085	1143
SA	2	47.2	0.439305	17000	8403	1270
SA	3	38.84	0.450124	17000	7622	1397
SA	3	37	0.556012	19000	14058	1524
SA	2	41.06	0.556254	19000	12613	1651
SA	2	33.16	0.508718	17000	11230	1778
SA	2	40.96	0.504783	17000	3119	1905
SA	2	28.38	0.644516	21000	17537	2032
SA	1	32.04	0.548834	18000	16279	2159
SA	2	45.42	0.513017	17000	5397	2286
SA	0	27.28	0.743748	23000	21641	2413
SA	1	33.24	1.02356	34000	22921	2540
TS	0	37.04	1.35586	62214	27816	127
TS	0	36.82	1.34593	61784	45595	254
TS	0	36.32	1.3567	62673	17209	381
TS	0	35.62	1.35031	62192	56160	508
TS	0	30.4	1.37816	62275	62222	635
TS	0	42.74	1.25654	62417	62369	762
TS	0	34.84	1.34626	61903	57823	889
TS	0	33.42	1.33401	62335	62283	1016
TS	0	37.48	1.37024	62201	55141	1143
TS	0	41.96	1.32666	61821	51804	1270
TS	0	38.98	1.31587	62334	58265	1397
TS	0	38.24	1.35883	62166	50071	1524
TS	0	35.24	1.36095	62386	59296	1651
TS	0	39.18	1.38548	62888	43665	1778
TS	0	35.44	1.32062	62169	59109	1905
TS	0	33.42	1.33557	61982	51862	2032
TS	0	34.84	1.36258	62090	46970	2159
TS	0	34.44	1.30218	62430	60375	2286
TS	0	34.3	1.34827	62090	37949	2413
TS	0	40.2	1.30904	62088	39879	2540

Table C.26: Measured data for the 70-node scale-free virtual topology.

Search Algorithm	Violations	Score	Time	Total Iterations	Iterations to Best	Random Seed
SA	3	45.6	0.497098	18000	8272	127
SA	3	41.76	0.519031	17000	12348	254
SA	4	36.2	0.635553	24000	20249	381
SA	3	39.26	0.473027	17000	13193	508
SA	2	50.22	0.486365	17000	3373	635
SA	1	44.9	0.501638	17000	10450	762
SA	2	37.06	0.520732	17000	14228	889
SA	3	37.56	0.533237	17000	11539	1016
SA	1	42.7	0.638813	20000	10409	1143
SA	3	36.06	0.555253	20000	11314	1270
SA	2	45.1	0.575484	21000	8679	1397
SA	2	46.84	0.50003	17000	8401	1524
SA	3	42.24	0.468757	17000	13537	1651
SA	1	41.4	0.644187	21000	12443	1778
SA	2	39.58	0.803749	27000	20414	1905
SA	2	32.2	0.702444	23000	17416	2032
SA	3	42.02	0.595761	22000	10379	2159
SA	2	43.46	0.585765	20000	14369	2286
SA	1	38.5	0.598356	19000	15195	2413
SA	3	37.86	0.479162	17000	13385	2540
TS	0	46.08	1.54361	72369	70304	127
TS	0	39.34	1.57681	72646	70576	254
TS	0	41.74	1.55142	72210	63168	381
TS	0	46.12	1.51279	72992	69940	508
TS	1	38.82	1.53314	72332	63214	635
TS	0	46.42	1.57858	72092	62052	762
TS	0	39.6	1.57274	72528	68490	889
TS	0	46.26	1.57332	72219	63149	1016
TS	0	41.14	1.55545	72532	66440	1143
TS	1	46.26	1.56891	72226	55080	1270
TS	0	40.48	1.57411	72341	72329	1397
TS	0	42.76	1.55832	71912	62779	1524
TS	1	40.64	1.54267	72351	62238	1651
TS	1	44.58	1.54797	72186	65111	1778
TS	1	40.72	1.55638	72100	70026	1905
TS	0	41.96	1.54319	72209	72164	2032
TS	0	46.84	1.56701	72290	57175	2159
TS	1	36.94	1.56821	72608	60442	2286
TS	0	38.32	1.57567	72531	67446	2413
TS	1	43.78	1.46738	73228	69167	2540

Table C.27: Measured data for the 80-node scale-free virtual topology.

Search Algorithm	Violations	Score	Time	Total Iterations	Iterations to Best	Random Seed
SA	6	49.52	0.420093	17000	15056	127
SA	4	52.44	0.489444	18000	11460	254
SA	3	54.72	0.528759	17000	3714	381
SA	7	53.88	0.40347	17000	3222	508
SA	5	50.12	0.492151	18000	10388	635
SA	5	56.04	0.415085	18000	12974	762
SA	4	38.18	0.662515	24000	19977	889
SA	4	52.62	0.372253	17000	7060	1016
SA	5	56.64	0.469419	17000	8098	1143
SA	3	49.44	0.67288	23000	7452	1270
SA	4	48.5	0.769008	27000	24646	1397
SA	4	48.96	0.488941	17000	12338	1524
SA	5	57.14	0.381558	17000	4761	1651
SA	6	46.36	0.466277	17000	10404	1778
SA	3	47.48	0.58933	22000	17538	1905
SA	5	51.22	0.415467	17000	10857	2032
SA	5	52.26	0.496941	17000	5364	2159
SA	4	51.08	0.460168	19000	13986	2286
SA	4	48.54	0.457406	17000	7331	2413
SA	4	48.18	0.620873	25000	22607	2540
TS	1	47.68	1.70819	82661	77605	127
TS	1	54.3	1.725	82519	38047	254
TS	1	40.74	1.79482	82165	82091	381
TS	0	54.8	1.77081	82578	65341	508
TS	2	48.42	1.76839	82815	38305	635
TS	1	53.72	1.75585	82680	82678	762
TS	2	52.3	1.78074	82604	70551	889
TS	2	52.48	1.37517	83711	29405	1016
TS	2	52.16	1.77769	82416	56252	1143
TS	0	53.56	1.75559	82519	41149	1270
TS	1	46.96	1.77697	82463	67380	1397
TS	3	48.92	1.73381	82653	79563	1524
TS	2	50.34	1.63688	83992	77947	1651
TS	1	51.16	1.78583	82380	81349	1778
TS	0	51.86	1.79137	82156	67056	1905
TS	3	44.38	1.75445	82411	70243	2032
TS	1	57.72	1.66872	82675	77561	2159
TS	1	56.84	1.81485	83363	83330	2286
TS	1	42.98	1.74539	82194	67050	2413
TS	1	53.46	1.75513	82502	75396	2540

Table C.28: Measured data for the 90-node scale-free virtual topology.

Search Algorithm	Violations	Score	Time	Total Iterations	Iterations to Best	Random Seed
SA	6	48.72	0.513016	18360	8110	127
SA	4	45	0.954106	31320	19411	254
SA	3	57.88	0.652337	22680	16239	381
SA	6	50.34	0.540788	18360	12978	508
SA	5	48.66	0.622175	20520	16570	635
SA	6	54.46	0.534492	18360	12003	762
SA	6	67.84	0.559594	18360	10831	889
SA	5	50.58	0.51954	18360	13216	1016
SA	6	56	0.6034	19440	15783	1143
SA	4	46.72	0.625533	20520	17330	1270
SA	3	63.8	0.506951	18360	5952	1397
SA	5	66.26	0.540455	18360	7786	1524
SA	4	54	0.978641	31320	19559	1651
SA	1	60.24	0.674061	21600	14876	1778
SA	3	52.1	0.663737	20520	17413	1905
SA	7	52.86	0.755831	28080	18445	2032
SA	6	60.22	0.5403	18360	10612	2159
SA	6	52.84	0.529508	18360	12284	2286
SA	6	59.1	0.564967	18360	13642	2413
SA	2	46.54	0.800372	25920	24483	2540
TS	0	53.82	2.10835	99616	94174	127
TS	0	59.84	2.1941	99470	98398	254
TS	0	56.42	2.13305	99567	88677	381
TS	3	54.86	2.09089	100641	80950	508
TS	1	50.52	2.09752	99745	95380	635
TS	0	51.08	2.16547	99524	96252	762
TS	3	57.74	2.10029	99327	9931	889
TS	0	47.26	2.1665	99717	81167	1016
TS	0	54.36	2.11994	99592	97430	1143
TS	1	54.36	2.15804	99711	92076	1270
TS	0	51.86	2.18963	99863	99847	1397
TS	0	52.2	2.22353	99982	98885	1524
TS	0	59.4	2.15442	99419	95067	1651
TS	0	54.3	2.1916	99668	97485	1778
TS	0	58.56	2.16382	100140	97974	1905
TS	1	52.72	2.12238	100098	96804	2032
TS	1	50.42	2.12168	99644	99619	2159
TS	2	59.08	2.12974	99727	95364	2286
TS	0	52.52	2.14279	99457	74421	2413
TS	4	51.06	2.04208	99742	98553	2540

Table C.29: Measured data for the 100-node scale-free virtual topology.

Search Algorithm	Violations	Score	Time	Total Iterations	Iterations to Best	Random Seed
SA	6	67.3	0.611924	20400	8207	127
SA	7	61.5	0.665526	24000	20644	254
SA	7	58.96	0.567272	21600	16979	381
SA	6	63.68	0.554089	20400	12051	508
SA	5	61.26	0.666372	22800	14868	635
SA	7	55.66	0.688471	24000	21496	762
SA	5	59.56	0.617178	20400	17066	889
SA	8	57.02	0.731403	25200	20525	1016
SA	8	60.06	0.631834	20400	13957	1143
SA	8	70.08	0.577854	20400	6770	1270
SA	7	69.98	0.888834	31200	16061	1397
SA	4	62.38	0.780139	27600	16249	1524
SA	7	51.04	1.02937	33600	31289	1651
SA	5	61.76	0.741565	25200	19894	1778
SA	5	59.98	0.582623	20400	12343	1905
SA	6	68.3	0.489472	20400	11254	2032
SA	8	63.02	0.612059	20400	8400	2159
SA	6	64.48	0.690803	22800	13799	2286
SA	5	59.06	0.728831	26400	17226	2413
SA	7	74.96	0.616096	20400	5101	2540
TS	0	61.68	2.74261	122816	114399	127
TS	1	62.26	2.70789	122339	119908	254
TS	1	61.4	2.58797	122856	120387	381
TS	1	52.06	2.71086	123113	112225	508
TS	2	50.3	2.62717	123153	72348	635
TS	1	55.8	2.62938	123160	114706	762
TS	0	60.14	2.71259	122703	117810	889
TS	0	53.16	2.66097	122545	121281	1016
TS	0	54.98	2.68444	122705	119074	1143
TS	2	66.86	2.58262	122727	99844	1270
TS	3	55.48	2.69744	122814	117964	1397
TS	0	62.9	2.7349	122622	99684	1524
TS	1	60.7	2.57555	122471	48839	1651
TS	0	61.36	2.69994	122854	109575	1778
TS	0	60.7	2.67553	122923	72311	1905
TS	2	59.72	2.58707	122741	121479	2032
TS	1	63.46	2.69013	122484	106812	2159
TS	3	59.82	2.74178	122697	115478	2286
TS	0	60.88	2.55702	122819	97471	2413
TS	0	59.76	2.61843	122805	102304	2540

Table C.30: Measured data for the 200-node scale-free virtual topology.

Search Algorithm	Violations	Score	Time	Total Iterations	Iterations to Best	Random Seed
SA	7	40.7	0.057986	1000	200	127
SA	8	41.16	0.057906	1000	200	254
SA	6	40.44	0.056796	1000	398	381
SA	7	40.9	0.057653	1000	289	508
SA	8	41.36	0.058151	1000	561	635
SA	12	43.2	0.058373	1000	992	762
SA	10	42.28	0.057001	1000	223	889
SA	6	40.44	0.057125	1000	481	1016
SA	6	40.44	0.057929	1000	320	1143
SA	7	40.9	0.057978	1000	668	1270
SA	10	42.28	0.057854	1000	253	1397
SA	6	40.24	0.057908	1000	200	1524
SA	10	42.28	0.057371	1000	240	1651
SA	7	40.9	0.057589	1000	230	1778
SA	8	41.36	0.05798	1000	753	1905
SA	9	41.62	0.057268	1000	200	2032
SA	9	41.82	0.057854	1000	648	2159
SA	8	41.16	0.056916	1000	390	2286
SA	9	41.82	0.058078	1000	987	2413
SA	8	41.16	0.056065	1000	200	2540
TS	2	38.6	5.72312	204059	144649	127
TS	3	39.06	5.71764	204058	8564	254
TS	2	38.6	5.73765	205010	92678	381
TS	2	38.6	5.71108	204088	28919	508
TS	2	38.6	5.72768	205128	21311	635
TS	2	38.6	5.73368	205033	80678	762
TS	2	38.6	5.73654	204106	37847	889
TS	1	38.14	5.74624	204844	83677	1016
TS	2	38.6	5.70229	203589	70374	1143
TS	2	38.6	5.76727	204402	191369	1270
TS	2	38.6	5.69313	202781	84191	1397
TS	2	38.6	5.7775	204733	63577	1524
TS	2	38.6	5.6828	202958	103444	1651
TS	2	38.6	5.77476	205084	147032	1778
TS	2	38.6	5.74513	204805	9431	1905
TS	2	38.6	5.75108	205138	41619	2032
TS	3	38.86	5.75468	203773	125913	2159
TS	2	38.6	5.76359	205215	177314	2286
TS	2	38.6	5.74105	203898	10568	2413
TS	2	38.6	5.73618	203016	17421	2540

Table C.31: Measured data for the 300-node scale-free virtual topology.

Search Algorithm	Violations	Score	Time	Total Iterations	Iterations to Best	Random Seed
SA	5	57.1	0.071714	1000	684	127
SA	10	59.4	0.071729	1000	909	254
SA	6	57.56	0.070662	1000	328	381
SA	4	56.64	0.07243	1000	581	508
SA	8	58.48	0.072385	1000	899	635
SA	9	58.74	0.070862	1000	300	762
SA	6	57.56	0.072275	1000	466	889
SA	3	56.18	0.071965	1000	649	1016
SA	11	59.86	0.07122	1000	800	1143
SA	4	56.44	0.071268	1000	300	1270
SA	5	57.1	0.071714	1000	478	1397
SA	4	56.64	0.071307	1000	905	1524
SA	6	57.56	0.073045	1000	835	1651
SA	4	56.64	0.073098	1000	696	1778
SA	14	61.24	0.070903	1000	679	1905
SA	9	58.94	0.071599	1000	532	2032
SA	6	57.36	0.07207	1000	300	2159
SA	7	57.82	0.070608	1000	369	2286
SA	13	60.78	0.072097	1000	980	2413
SA	6	57.56	0.071121	1000	307	2540
TS	0	54.8	10.1945	307079	3353	127
TS	0	54.8	10.139	304991	54570	254
TS	1	55.26	10.1748	306371	44746	381
TS	0	54.8	10.2134	307284	23856	508
TS	1	55.26	10.102	303167	18320	635
TS	0	54.8	10.1604	307238	9550	762
TS	1	55.26	10.1971	307341	24882	889
TS	1	55.26	10.1847	305331	18785	1016
TS	0	54.8	10.1227	304081	83788	1143
TS	1	55.26	10.1564	307086	109343	1270
TS	1	55.26	10.1817	306420	101254	1397
TS	0	54.8	10.2027	305199	86951	1524
TS	0	54.8	10.1815	306068	160912	1651
TS	1	55.26	10.1653	307047	69822	1778
TS	1	55.26	10.2066	306892	47500	1905
TS	0	54.8	10.1369	303875	246847	2032
TS	1	55.26	10.2088	306252	143482	2159
TS	1	55.26	10.1404	306783	26327	2286
TS	1	55.26	10.1415	305155	295794	2413
TS	1	55.26	10.0248	304267	46930	2540

Table C.32: Measured data for the 400-node scale-free virtual topology.

Search Algorithm	Violations	Score	Time	Total Iterations	Iterations to Best	Random Seed
SA	27	84.3	0.085365	1000	469	127
SA	21	81.54	0.0897	1000	574	254
SA	18	79.96	0.087815	1000	400	381
SA	13	77.86	0.088121	1000	623	508
SA	14	78.32	0.094667	1000	438	635
SA	20	81.08	0.088723	1000	944	762
SA	15	78.78	0.087469	1000	540	889
SA	12	77.4	0.088364	1000	887	1016
SA	18	79.96	0.089366	1000	990	1143
SA	18	80.16	0.087504	1000	436	1270
SA	15	78.78	0.088011	1000	885	1397
SA	19	80.62	0.088566	1000	982	1524
SA	18	80.16	0.085486	1000	678	1651
SA	18	80.16	0.091757	1000	583	1778
SA	38	89.36	0.086983	1000	495	1905
SA	19	80.62	0.087379	1000	514	2032
SA	13	77.86	0.087268	1000	405	2159
SA	20	81.08	0.088346	1000	405	2286
SA	19	80.62	0.090379	1000	998	2413
SA	16	79.24	0.086838	1000	422	2540
TS	4	73.72	14.9618	406684	200504	127
TS	4	73.72	14.998	407827	297752	254
TS	6	74.64	15.1073	408585	37043	381
TS	4	73.72	14.9784	405077	390088	508
TS	6	74.64	15.0062	408672	189466	635
TS	4	73.72	14.9746	406850	229656	762
TS	5	74.18	14.8776	404737	83279	889
TS	5	74.18	15.0336	408020	299915	1016
TS	5	74.18	15.0147	407776	112404	1143
TS	5	74.18	14.9346	404364	213192	1270
TS	5	74.18	15.096	408829	57323	1397
TS	5	74.18	15.1133	408710	92260	1524
TS	5	74.18	15	407647	47180	1651
TS	4	73.72	15.072	407817	234647	1778
TS	5	74.18	15.1398	408726	176474	1905
TS	6	74.64	14.9162	406725	134452	2032
TS	4	73.72	15.0872	408750	116361	2159
TS	5	74.18	15.1005	408213	225025	2286
TS	4	73.72	15.0864	408960	386932	2413
TS	4	73.72	14.9418	405928	65498	2540

Table C.33: Measured data for the 500-node scale-free virtual topology.

Search Algorithm	Violations	Score	Time	Total Iterations	Iterations to Best	Random Seed
SA	14	94.16	0.107073	1000	530	127
SA	14	94.16	0.108007	1000	712	254
SA	20	96.92	0.107421	1000	723	381
SA	23	98.3	0.107198	1000	829	508
SA	23	98.3	0.108	1000	649	635
SA	16	95.08	0.106049	1000	522	762
SA	35	103.62	0.106182	1000	500	889
SA	24	98.56	0.109655	1000	897	1016
SA	19	96.46	0.107887	1000	922	1143
SA	23	98.1	0.107043	1000	500	1270
SA	25	99.22	0.109658	1000	795	1397
SA	19	96.46	0.107713	1000	589	1524
SA	22	97.84	0.106441	1000	522	1651
SA	14	94.16	0.107931	1000	521	1778
SA	27	100.14	0.106386	1000	524	1905
SA	15	94.62	0.107295	1000	579	2032
SA	20	96.92	0.106198	1000	995	2159
SA	20	96.92	0.107951	1000	896	2286
SA	19	96.46	0.108255	1000	901	2413
SA	20	96.92	0.106058	1000	588	2540
TS	3	89.1	19.971	506669	488635	127
TS	2	88.64	20.1655	508665	371547	254
TS	4	89.36	19.9679	506259	176939	381
TS	3	89.1	20.1247	507767	406729	508
TS	3	89.1	20.0462	506592	264324	635
TS	4	89.56	20.0008	505011	114554	762
TS	3	89.1	19.9808	504386	179059	889
TS	4	89.36	20.1438	508776	294574	1016
TS	2	88.64	20.0072	507552	411490	1143
TS	3	88.9	20.0879	504704	450675	1270
TS	4	89.56	19.9235	506208	122816	1397
TS	3	89.1	20.0382	507828	367694	1524
TS	3	89.1	19.9563	507567	376425	1651
TS	3	89.1	20.038	506828	450768	1778
TS	2	88.64	19.9878	505290	233941	1905
TS	3	89.1	20.0676	507728	151303	2032
TS	3	89.1	20.0488	507156	413065	2159
TS	3	89.1	20.045	506819	277658	2286
TS	3	89.1	19.9986	504975	255724	2413
TS	3	89.1	20.0398	506422	191087	2540

Table C.34: Measured data for the 600-node scale-free virtual topology.

Search Algorithm	Violations	Score	Time	Total Iterations	Iterations to Best	Random Seed
SA	19	113.26	0.133441	1000	771	127
SA	17	112.34	0.13641	1000	798	254
SA	8	108.2	0.132592	1000	808	381
SA	10	108.92	0.133616	1000	600	508
SA	19	113.26	0.133757	1000	991	635
SA	12	110.04	0.132746	1000	864	762
SA	15	111.22	0.133001	1000	600	889
SA	12	109.84	0.134011	1000	600	1016
SA	11	109.58	0.134054	1000	745	1143
SA	19	113.26	0.132764	1000	994	1270
SA	18	112.8	0.133008	1000	806	1397
SA	15	111.22	0.133649	1000	647	1524
SA	20	113.72	0.131982	1000	634	1651
SA	19	113.26	0.134514	1000	832	1778
SA	12	109.84	0.13295	1000	600	1905
SA	12	110.04	0.135335	1000	730	2032
SA	15	111.42	0.133213	1000	706	2159
SA	21	114.18	0.134622	1000	875	2286
SA	13	110.5	0.132144	1000	738	2413
SA	11	109.58	0.13456	1000	840	2540
TS	2	105.44	26.3692	609448	23644	127
TS	2	105.44	26.305	607522	15659	254
TS	1	104.98	26.2713	607067	115705	381
TS	0	104.52	26.275	606270	66549	508
TS	1	104.98	26.2948	608194	585173	635
TS	0	104.52	26.1788	605547	523533	762
TS	1	104.98	26.3057	606287	181905	889
TS	1	104.98	26.1431	605390	595421	1016
TS	0	104.52	26.3541	606992	430818	1143
TS	0	104.52	26.326	607417	334152	1270
TS	0	104.52	26.21	605505	478379	1397
TS	0	104.52	26.3316	610436	512409	1524
TS	2	105.44	26.2669	605041	118497	1651
TS	1	104.98	26.1699	605705	443548	1778
TS	1	104.98	26.3273	606865	353592	1905
TS	1	104.98	26.2546	606766	513695	2032
TS	0	104.52	26.3704	609524	492412	2159
TS	0	104.52	26.2911	606439	262168	2286
TS	2	105.44	26.1952	607394	260124	2413
TS	1	104.98	26.2416	605522	358397	2540

Table C.35: Measured data for the 700-node scale-free virtual topology.

Search Algorithm	Violations	Score	Time	Total Iterations	Iterations to Best	Random Seed
SA	28	135.96	0.164922	1000	704	127
SA	40	141.48	0.16471	1000	904	254
SA	27	135.5	0.163599	1000	821	381
SA	42	142.4	0.161642	1000	972	508
SA	37	140.1	0.164278	1000	776	635
SA	30	136.88	0.16639	1000	917	762
SA	27	135.5	0.165944	1000	845	889
SA	28	135.96	0.166433	1000	930	1016
SA	22	133.2	0.167807	1000	787	1143
SA	22	133.2	0.165342	1000	744	1270
SA	35	139.18	0.165069	1000	807	1397
SA	17	130.7	0.167739	1000	858	1524
SA	36	139.44	0.164228	1000	742	1651
SA	31	137.14	0.167513	1000	700	1778
SA	21	132.74	0.164901	1000	859	1905
SA	35	139.18	0.164886	1000	758	2032
SA	38	140.36	0.165678	1000	984	2159
SA	42	142.4	0.162082	1000	957	2286
SA	34	138.72	0.166328	1000	850	2413
SA	26	134.84	0.16497	1000	700	2540
TS	1	123.54	33.6259	706117	439853	127
TS	4	124.92	33.7103	706645	220181	254
TS	4	124.92	33.5836	707854	582767	381
TS	4	124.92	33.759	710245	496021	508
TS	2	124	33.7591	708124	240650	635
TS	1	123.54	33.7774	709647	693702	762
TS	3	124.46	33.6124	706035	464842	889
TS	4	124.92	33.5691	707167	230626	1016
TS	4	124.92	33.5806	706659	183044	1143
TS	3	124.46	33.6454	707762	134071	1270
TS	1	123.54	33.6609	704864	656884	1397
TS	5	125.38	33.5367	705253	685253	1524
TS	4	124.92	33.6448	708870	413634	1651
TS	2	124	33.6771	706854	562728	1778
TS	1	123.54	33.6831	707637	292258	1905
TS	3	124.46	33.615	706603	403480	2032
TS	3	124.46	33.7644	707289	183687	2159
TS	3	124.46	33.5978	706145	366822	2286
TS	2	124	33.6102	706368	282006	2413
TS	2	124	33.6444	708052	450855	2540

Table C.36: Measured data for the 800-node scale-free virtual topology.

Search Algorithm	Violations	Score	Time	Total Iterations	Iterations to Best	Random Seed
SA	39	156.54	0.190656	1000	919	127
SA	25	149.9	0.194599	1000	800	254
SA	28	151.28	0.196408	1000	800	381
SA	31	152.86	0.194562	1000	898	508
SA	27	151.02	0.194641	1000	826	635
SA	21	148.06	0.196925	1000	800	762
SA	21	148.26	0.198724	1000	939	889
SA	22	148.72	0.194735	1000	947	1016
SA	35	154.7	0.195318	1000	993	1143
SA	35	154.7	0.198705	1000	943	1270
SA	28	151.28	0.197155	1000	800	1397
SA	23	149.18	0.195717	1000	928	1524
SA	26	150.56	0.197065	1000	941	1651
SA	43	158.38	0.195542	1000	815	1778
SA	15	145.3	0.193735	1000	800	1905
SA	18	146.88	0.197165	1000	870	2032
SA	19	147.34	0.197419	1000	847	2159
SA	48	160.68	0.190136	1000	846	2286
SA	23	149.18	0.19772	1000	808	2413
SA	24	149.64	0.196298	1000	837	2540
TS	0	138.6	40.5585	807017	640893	127
TS	1	139.06	40.4271	807098	164503	254
TS	0	138.6	40.4849	806671	789729	381
TS	0	138.6	40.43	805762	388539	508
TS	1	138.86	40.2601	804562	762545	635
TS	0	138.4	40.456	807043	319596	762
TS	0	138.6	40.5264	808175	557916	889
TS	0	138.4	40.5223	809357	465029	1016
TS	1	138.86	40.4962	807051	534832	1143
TS	0	138.4	40.468	807684	363224	1270
TS	0	138.6	40.4735	809587	746563	1397
TS	0	138.6	40.4477	805334	447978	1524
TS	0	138.6	40.5848	807328	466014	1651
TS	1	138.86	40.4037	806224	656110	1778
TS	0	138.4	40.4512	806374	706308	1905
TS	1	139.06	40.4062	806691	115977	2032
TS	0	138.6	40.3669	806153	786283	2159
TS	1	139.06	40.4225	805764	184176	2286
TS	1	139.06	40.5068	806464	344986	2413
TS	0	138.6	40.6198	808639	183995	2540

Table C.37: Measured data for the 900-node scale-free virtual topology.

Search Algorithm	Violations	Score	Time	Total Iterations	Iterations to Best	Random Seed
SA	41	173.94	0.229077	1000	955	127
SA	30	168.68	0.231913	1000	900	254
SA	24	165.92	0.232653	1000	900	381
SA	47	176.5	0.230813	1000	900	508
SA	40	173.48	0.230753	1000	904	635
SA	19	163.62	0.232793	1000	900	762
SA	44	175.32	0.232491	1000	936	889
SA	29	168.42	0.230677	1000	922	1016
SA	11	159.94	0.233863	1000	900	1143
SA	40	173.48	0.234312	1000	975	1270
SA	37	172.1	0.232686	1000	988	1397
SA	44	175.12	0.229167	1000	900	1524
SA	36	171.44	0.226848	1000	900	1651
SA	43	174.66	0.233356	1000	900	1778
SA	29	168.22	0.232517	1000	900	1905
SA	32	169.6	0.227636	1000	900	2032
SA	33	170.06	0.230073	1000	900	2159
SA	30	168.88	0.229578	1000	986	2286
SA	59	182.02	0.23298	1000	900	2413
SA	24	166.12	0.233371	1000	967	2540
TS	3	156.46	48.2428	909109	420685	127
TS	2	156	48.2593	907201	506861	254
TS	3	156.46	48.413	908280	483879	381
TS	1	155.54	48.377	906913	658674	508
TS	2	155.8	48.2886	905660	382212	635
TS	1	155.54	48.4007	907219	770154	762
TS	1	155.54	48.2869	906946	323362	889
TS	3	156.46	48.4013	909224	421763	1016
TS	2	156	48.3814	907424	558078	1143
TS	0	155.08	48.3772	907501	650243	1270
TS	2	156	48.4747	909574	396079	1397
TS	2	156	48.2974	907594	459146	1524
TS	1	155.54	48.2819	907172	525825	1651
TS	3	156.46	48.2828	906703	523338	1778
TS	2	156	48.3148	907260	409783	1905
TS	1	155.54	48.3515	905371	342795	2032
TS	1	155.54	48.317	906834	293223	2159
TS	1	155.54	48.2878	907840	648605	2286
TS	0	155.08	48.3519	907206	735020	2413
TS	2	156	48.4663	908933	295338	2540

Table C.38: Measured data for the 1000-node scale-free virtual topology.

Search Algorithm	Violations	Score	Time	Total Iterations	Iterations to Best	Random Seed
SA	26	184.6	0.278927	1000	1000	127
SA	47	194.26	0.275766	1000	1000	254
SA	37	189.66	0.278949	1000	1000	381
SA	50	195.64	0.27478	1000	1000	508
SA	45	193.34	0.278408	1000	1000	635
SA	35	188.74	0.276473	1000	1000	762
SA	27	185.06	0.275374	1000	1000	889
SA	34	188.28	0.278871	1000	1000	1016
SA	38	190.12	0.27748	1000	1000	1143
SA	43	192.42	0.274125	1000	1000	1270
SA	31	186.9	0.279192	1000	1000	1397
SA	30	186.44	0.273714	1000	1000	1524
SA	31	186.9	0.27356	1000	1000	1651
SA	56	198.4	0.279626	1000	1000	1778
SA	32	187.36	0.273459	1000	1000	1905
SA	59	199.78	0.269005	1000	1000	2032
SA	45	193.94	0.273826	1000	998	2159
SA	38	190.12	0.274202	1000	1000	2286
SA	31	186.9	0.276613	1000	1000	2413
SA	39	190.58	0.276491	1000	1000	2540
TS	1	173.3	58.1444	1008020	761827	127
TS	1	173.1	58.1174	1006467	863365	254
TS	0	172.84	58.0631	1006136	946120	381
TS	2	173.76	57.9998	1005046	658814	508
TS	0	172.84	58.1346	1007684	997658	635
TS	2	173.76	57.9416	1005815	294233	762
TS	1	173.3	58.0456	1006903	596540	889
TS	0	172.84	58.0119	1006824	446400	1016
TS	1	173.3	58.0682	1006666	631297	1143
TS	0	172.84	58.2383	1005888	950853	1270
TS	1	173.3	58.0249	1005736	791527	1397
TS	2	173.56	58.004	1005874	984888	1524
TS	0	172.84	58.0754	1006855	398285	1651
TS	0	172.84	58.1183	1006686	789474	1778
TS	0	172.84	58.1102	1005732	520249	1905
TS	4	174.68	58.1143	1007698	826512	2032
TS	0	172.84	58.0984	1008157	862007	2159
TS	0	172.64	58.1433	1006590	530101	2286
TS	2	173.56	58.0591	1005572	603274	2413
TS	1	173.3	57.9541	1005853	639522	2540

Bibliography

1. Albert, Reka, Hawoong Jeong, and Albert-Laszlo Barabasi. "Error and Attack Tolerance of Complex Networks". *Nature*, 406(6794):378–382, Jul 2000. URL `http://dx.doi.org/10.1038/35019019`.

2. Baldwin, Rusty O. "Fundamentals of Performance Analysis and Experimental Design". CSCE 554 Course Lecture Slides, August 2006.

3. Barabasi, Albert-Laszlo and Reka Albert. "Emergence of Scaling in Random Networks". *Science*, 286(5439):509–512, Oct 1999. URL `http://arxiv.org/abs/cond-mat/9910332`.

4. Barabasi, Albert-Laszlo and Eric Bonabeau. "Scale-free Networks". *Scientific American*, 288(5):60, May 2003.

5. Battiti, Robert and Alan Bertossi. "Greedy, Prohibition, and Reactive Heuristics for Graph Partitioning". *IEEE Transactions on Computers*, 48(4):361–385, April 1999.

6. Chen, Clement C. "Anatomy of Network-Centric Warfare". *Armed Forces Communications and Electronics Association Signal Magazine*, 57(12):47, August 2003.

7. Dekker, Anthony H. "Network Topology and Military Performance". *Proceedings of MODSIM 2005 International Congress on Modelling and Simulation*, 2174–2180, December 2005. URL `http://www.mssanz.org.au/modsim05/`.

8. Dreo, Johann, Alain Petrowski, Patrick Siarry, and Eric Taillard. *Metaheuristics for Hard Optimization*. Springer-Verlag Berlin Heidelburg, 2006.

9. Erdos, Pal and Alfred Renyi. "On the Evolution of Random Graphs". *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 5:17–61, 1960.

10. Glover, Fred and Manuel Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.

11. Guruprasad, Shashikiran, Rob Ricci, and Jay Lepreau. "Integrated Network Experimentation using Simulation and Emulation". *In Proceedings of TridentCom 2005*, February 2005. URL `http://www.emulab.net/pubs.php3`.

12. Guruprasad, Shashikiran B. *Issues in Integrated Network Experimentation Using Simulation and Emulation*. Master's thesis, The University of Utah Graduate School, August 2005.

13. Harris, Ryan. *C.O.R.E. CyberOpeRations Emulator Getting Started Guide*. Air Force Institute of Technology, March 2006.

14. Hibler, Mike, Robert Ricci, Leigh Stoller, Jonathon Duerig, Shashi Guruprasad, Tim Stack, Kirk Webb, and Jay Lepreau. *Feedback-directed Virtualization Techniques for Scalable Network Experimentation*. Technical Note FTN-2004-02, University of Utah Flux Group, May 2004. URL `http://www.cs.utah.edu/flux/papers/virt-ftn2004-02-base.html`.

15. Huffaker, Bradley, Evi Nemeth, and K Claffy. "Otter: A General-purpose Network Visualization Tool". *In Proceedings of INET 1999*. June 1999.

16. Jarvis, David A. *A Methodology for Analyzing Complex Military Command and Control (C2) Networks*. Technical report, Alidade Incorporated, 31 Bridge Street Newport, Rhode Island 02840, 2005. URL `http://www.alidade.net/recent_research/`.

17. Karypis, George and Vipin Kumar. "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs". *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998. URL `http://link.aip.org/link/?SCE/20/359/1`.

18. Li, Lun, David Alderson, Reiko Tanaka, John C. Doyle, and Walter Willinger. *Towards a Theory of Scale-Free Graphs: Definition, Properties, and Implications*. Technical Report CIT-CDS-04-006, California Institute of Technology, Pasadena, CA, USA, Oct 2005.

19. Medina, Alberto, Anukool Lakhina, Ibrahim Matta, and John Byers. *BRITE: Universal Topology Generation from a User's Perspective*. Technical Report 2001-003, Boston University, 1 2001. URL `citeseer.ist.psu.edu/medina01brite.html`.

20. Nurminen, Jukka K. *Models and Algorithms for Network Planning Tools - Practical Experiences*. Technical report, Helsinki University of Technology, May 2003. URL `http://lib.hut.fi/Diss/2003/isbn9512265745/article6.pdf`.

21. Pakstas, Algirdas and Muhammad Azizur Rahman. "Incompatibiliy of Network Design and Simulation Tools and an Approach to Integration". *Proceedings of the 6th EPSRC Annual Postgraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting (EPSRC PGNet 2005)*, 529–535. Liverpool John Moores University, Liverpool, England, June 2005.

22. Pawlikowski, Krzysztof, Hae-Duck Joshua Jeong, and Jong-Suk Ruth Lee. "On Credibility of Simulation Studies of Telecommunication Networks". *IEEE Communications Magazine*, 40(1):132–139, Jan 2002.

23. Rahman, Md Azizur, Algirdas Pakstas, and Frank Zhigang Wang. "An Approach to Integration of Network Design and Simulation Tools". *Proceedings of the 8th International Conference on Telecommunications (ConTEL 2005)*, volume 1, 173–180. June 2005.

24. Ricci, Robert, Chris Alfeld, and Jay Lepreau. "A Solver for the Network Testbed Mapping Problem". *SIGCOMM Computer Commu-*

nication Review, 33(2):65–81, April 2003. ISSN 0146-4833. URL http://portal.acm.org/citation.cfm?id=956988.

25. Robertazzi, Thomas G. *Computer Networks and Systems*. New York: Springer-Verlag, 3rd edition, 2000.

26. Silicon Graphics Computer Systems Incorporated. "Multimap". Standard Technical Library Entry, December 2006. URL http://www.sgi.com/tech/stl/Multimap.html.

27. Silicon Graphics Computer Systems Incorporated. "Sorted Associative Container". Standard Technical Library Entry, December 2006. URL http://www.sgi.com/tech/stl/SortedAssociativeContainer.html.

28. Strogatz, Steven H. "Exploring Complex Networks". *Nature*, 410(6825):268–276, March 2001. URL http://dx.doi.org/10.1038/35065725.

29. The University of Utah. "Emulab — Network Emulation Testbed Homepage", January 2007. URL http://www.emulab.net/.

30. The University of Utah and the Flux Group. "Assign about.txt". Text File Included With Emulab Environment Source Code, July 2002.

31. The University of Utah and the Flux Group. "Assign Readme". Text File Included With Emulab Environment Source Code, July 2002.

32. Wang, Xiao F. and Guanrong Chen. "Complex Networks: Small-world, Scale-free and Beyond". *IEEE Circuits and Systems Magazine*, 3(1):6–20, First Quarter 2003. ISSN 1531-636X. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1228503.

33. White, Brian, Jay Lepreau, Leigh Stoller, Robert Ricci, Shashi Guruprasad, Mac Newbold, Mike Hibler, Chad Barb, and Abhijeet Joglekar. "An Integrated Experimental Environment for Distributed Systems and Networks". *ACM SIGOPS Operating Systems Review*, 36(SI):255–270, 2002. ISSN 0163-5980.

34. Wikipedia. "Complex network — Wikipedia, The Free Encyclopedia", 2006. URL http://en.wikipedia.org/w/index.php?title=Complex_network.

35. Wikipedia. "Metaheuristic — Wikipedia, The Free Encyclopedia", 2006. URL http://en.wikipedia.org/w/index.php?title=Metaheuristic.

36. Wikipedia. "Scale-free network — Wikipedia, The Free Encyclopedia", 2006. URL http://en.wikipedia.org/w/index.php?title=Scale-free_network.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*
22–03–2007

2. REPORT TYPE
Master's Thesis

3. DATES COVERED *(From — To)*
Sept 2005 — Mar 2007

4. TITLE AND SUBTITLE
Use of Tabu Search in a Solver to Map Complex Networks onto Emulab Testbeds

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Jason E. MacDonald, Capt, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology
Graduate School of Engineering and Management
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GCE/ENG/07-07

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Communication Agency/Dynamic Network Analysis Division
Mr. Scott Gardner
Scott AFB, IL 62225
DSN 779-6794
Commercial (618) 229-6794

10. SPONSOR/MONITOR'S ACRONYM(S)
AFCA/ENAN

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
Approval for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The University of Utah's solver for the testbed mapping problem uses a simulated annealing metaheuristic algorithm to map a researcher's experimental network topology onto available testbed resources. This research uses tabu search to find near-optimal physical topology solutions to user experiments consisting of scale-free complex networks. While simulated annealing arrives at solutions almost exclusively by chance, tabu search incorporates the use of memory and other techniques to guide the search towards good solutions. Both search algorithm are compared to determine whether tabu search can produce equal or higher quality solutions than simulated annealing in a shorter amount of time. It is assumed that all testbed resources remain available, and that hardware faults or another competing mapping process do not remove testbed resources while either search algorithm is executing. The results show that tabu search produces a higher proportion of valid solutions for 34 out of the 38 test networks than simulated annealing. For cases where a valid solution was found, tabu search executes more quickly for scale-free networks and networks with less than 100 nodes.

15. SUBJECT TERMS
testbed mapping problem, simulated annealing, tabu search, network emulation, emulation

16. SECURITY CLASSIFICATION OF: a. REPORT	b. ABSTRACT	c. THIS PAGE	17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
U	U	U	UU	144	Dr. Robert F. Mills (ENG)

19b. TELEPHONE NUMBER *(include area code)*
(937) 255–3636 x4527, Robert.Mills@afit.edu